PHY Interface
For the
PCI Express, SATA, USB 3.1, DisplayPort, and Converged IO Architectures

Version 5.0

Intellectual Property Disclaimer

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER INCLUDING ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION, OR SAMPLE.

A COPYRIGHT LICENSE IS HEREBY GRANTED TO REPRODUCE AND DISTRIBUTE THIS SPECIFICATION FOR INTERNAL USE ONLY. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY OTHER INTELLECTUAL PROPERTY RIGHTS IS GRANTED OR INTENDED HEREBY.

INTEL CORPORATION AND THE AUTHORS OF THIS SPECIFICATION DISCLAIM ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF PROPRIETARY RIGHTS, RELATING TO IMPLEMENTATION OF INFORMATION IN THIS DOCUMENT AND THE SPECIFICATION. INTEL CORPORATION AND THE AUTHORS OF THIS SPECIFICATION ALSO DO NOT WARRANT OR REPRESENT THAT SUCH IMPLEMENTATION(S) WILL NOT INFRINGE SUCH RIGHTS.

ALL SUGGESTIONS OR FEEDBACK RELATED TO THIS SPECIFICATION BECOME THE PROPERTY OF INTEL CORPORATION UPON SUBMISSION.

INTEL CORPORATION MAY MAKE CHANGES TO SPECIFICATIONS, PRODUCT DESCRIPTIONS, AND PLANS AT ANY TIME, WITHOUT NOTICE.

Notice: Implementations developed using the information provided in this specification may infringe the patent rights of various parties including the parties involved in the development of this specification. No license, express or implied, by estoppel or otherwise, to any intellectual property rights (including without limitation rights under any party's patents) are granted herein.

All product names are trademarks, registered trademarks, or service marks of their respective owners

Contributors

Jeff Morris	Jim Choate	Michelle Jen	Kaleb Ruof
Andy Martwick	Paul Mattos	Bruce Tennant	John Watkins
Brad Hosler	Dan Froelich	Quinn Devine	Jamie Johnston
Matthew Myers	Duane Quiet	Su Wei Lim	Todd Witter
Bob Dunstan	Hajime Nozaki	Hooi Kar Loo	Andrea Uguagliati
Saleem Mohammad	Peter Teng	Poh Thiam Teoh	Efraim Kugman
Sue Vining	Karthi Vadivelu	Sathyanarayanan Gopal	Daniel Resnick
Tadashi Iwasaki	Mineru Nishizawa	Siang Lin Tan	
Yoichi Iizuka	Takanori Saeki	Jake Li	
Rahman Ismail	Andrew Lillie	Zeeshan Sarwar	
Ben Graniello	Frank Kavanagh	Minxi Gao	

Dedicated to the memory of Brad Hosler, the impact of whose accomplishments made the Universal Serial Bus one of the most successful technology innovations of the Personal Computer era.

Table of Contents

Table of Figures

Table of Tables

1 Preface

1.1 Scope of this Revision

The PCI Express, SATA, USB, DisplayPort, and Converged IO PHY Interface Specification has definitions of all functional blocks and signals. This revision includes support for PCI Express implementations conforming to the PCI Express Base Specification, Revision 4.0, SATA implementations conforming to the SATA specification, revision 3.0, USB implementations conforming to the Universal Serial Bus Specification, Revision 3.1, DisplayPort implementations conforming to the DisplayPort 1.3 Specification, and Converged IO implementations conforming to the Converged IO Base Specification, Revision 1.0

1.2 Revision History

Revision Number	Date	Description
0.1	7/31/02	Initial Draft
0.5	8/16/02	Draft for industry review
0.6	10/4/02	Provides operational detail
0.7	11/4/02	Includes timing diagrams
0.8	11/22/02	More operational detail. Receiver detection sequence changed.
0.9	12/16/02	Minor updates. Solid enough for implementations to be finalized.
0.95	4/25/03	Updates to reflect 1.0a Base Spec. Added multilane suggestions.
1.00	6/19/03	Stable revision for implementation.
1.70	11/6/05	First pass at Gen. 2 PIPE
1.81	12/4/2005	Fixed up areas based on feedback.
1.86	2/27/2006	Fixed up more areas based on feedback. Added a section on how to handle CLKREQ#.
1.87	9/28/2006	Removed references to Compliance Rate determination. Added sections for TX Margining and Selectable De-emphasis. Fixed up areas (6.4) based on feedback.
1.90	3/24/2007	Minor updates, mostly editorial.
2.00	7/21/2007	Minor updates, stable revision for implementation.
2.7	12/31/2007	Initial draft of updates to support the USB specification, revision 3.0.
2.71	1/21/2008	Updates for SKP handling and USB SuperSpeed PHY power management.
2.75	2/8/08	Additional updates for SKP handling.
2.90	8/11/08	Added 32 bit data interface support for USB SuperSpeed mode, support for USB SuperSpeed mode receiver equalization training, and support for USB SuperSpeed mode compliance patterns that are not 8b/10b encoded. Solid enough for implementation architectures to be finalized.
3.0	3/11/09	Final update
4.0	4/5/11	Draft 1 update adding SATA.
4.0	4/13/11	Draft 3 update adding PCI Express 3.0 rev .9.
4.0	9/1/11	Draft 6 update adding updates based on PCI Express 3.0 rev .9 feedback.
4.1	12/7/11	Initial draft with per lane clocking option
4.1	12/12/11	Draft 2. Updates for initial review feedback and addition of several example timing diagrams for PCI Express 3.0 related signals.
4.1	5/21/12	Updated for Draft 2 feedback from various reviewers.
4.2	7/1/13	Added support for USB 3.1 – preliminary review release based on

		USB 3.1 specification revision .9
4.3	1/31/14	Added support for PTM (preliminary for review), L1 Substates (preliminary for review), and PCI Express 4.0 (preliminary rev .3).
4.4	11/28/16	Added support for PCIe RX margining and elastic buffer depth control over a message bus interface. Support for PCIe Nominal Empty elastic buffer mode. Gen4 updates: LocalLF/FS, LF/FS, Rate, PCLK rates. SRIS support. RXStandby for USB. L1 substate clarifications. General cleanup.
4.4.1	1/12/17	Removed "PCLK as an input" requirement for message bus. Added wording to allow PHY to choose whether to support L1 substate management via PowerDown[3:0] exclusively or via RxEIDetectDisable and TxCommonModeDisable.
5.0	11/2/17	Clarified that margin NAK is only required for unsupported voltage margin offset requests that are within PHY advertised range. Added support for 64-bit data width for PCIe SerDes only. Mapped all eligible legacy PIPE signals into message bus registers. Added support for a SerDes architecture. Added requirements for support of low pin count vs legacy PIPE interface and SerDes vs original PIPE architecture. Added support for Converged IO and DisplayPort. Recommendation that USB Nominal Empty Operation should use RxDataValid. Added EB Error recovery mechanism controlled via a register bit. Added RefClkRequired signal to indicate when the reference clock can be safely removed. Reformatted signal tables into separate input and output tables and added a new column indicating relevant protocols. General cleanup and clarifications.

2 Introduction

The **P**HY **I**nterface for the **P**CI **E**xpress, SATA, USB, DisplayPort and Converged IO Architectures (**PIPE**) is intended to enable the development of functionally equivalent PCI Express, SATA, USB, DisplayPort, and Converged IO PHY's. Such PHY's can be delivered as discrete IC's or as macrocells for inclusion in ASIC designs. The specification defines a set of PHY functions which must be incorporated in a PIPE compliant PHY, and it defines a standard interface between such a PHY and a Media Access Layer (MAC) & Link Layer ASIC. It is not the intent of this specification to define the internal architecture or design of a compliant PHY chip or macrocell. The PIPE specification is defined to allow various approaches to be used. Where possible the PIPE specification references the PCI Express base specification, SATA 3.0 Specification, USB 3.1 Specification, DisplayPort 1.3 specification or Converged IO 1.0 specification rather than repeating its content. In case of conflicts, the PCI-Express Base Specification, SATA 3.0 specification, USB 3.1 Specification, DisplayPort 1.3 specification, and Converge IO 1.0 specification shall supersede the PIPE spec.

This spec provides some information about how the MAC could use the PIPE interface for various LTSSM states, Link states and other protocols. This information should be viewed as ‘guidelines for’ or as ‘one way to implement’ base specification requirements. MAC implementations are free to do things in other ways as long as they meet the corresponding specification requirements.

One of the intents of the PIPE specification is to accelerate PCI Express endpoint, SATA device, USB device, and Converged IO device development. This document defines an interface to which ASIC and endpoint device vendors can develop. Peripheral and IP vendors will be able to develop and validate their designs, insulated from the high-speed and analog circuitry issues associated with the PCI Express, SATA, USB, DisplayPort, or Converged IO PHY interfaces, thus minimizing the time and risk of their development cycles.

The PIPE specification defines two clocking options for the interface. In the first alternative the PHY provides a clock (PCLK) that clocks the PIPE interface as an output. In the second alternative PCLK is provided to each lane of the PHY as an input. The alternative, where PCLK is provided to each lane of the PHY, was added in the 4.1 revision of the PIPE specification. It allows the controller or logic external to the PHY to more easily adjust timing of the PIPE interface to meet timing requirements for silicon implementations. A PHY is only required to support one of the timing alternatives. The two clocking options shall be referenced as “PCLK as PHY Output” and “PCLK as PHY Input” respectively. DisplayPort only supports the “PCLK as PHY Input” clocking option. Note: The current trend is to deprecate “PCLK as PHY Output” mode in a future revision of this specification.

Figure 2-1: Partitioning PHY Layer for PCI Express shows the partitioning described in this spec for the PCI Express Base Specification. Figure 2-2 shows the partitioning described in this spec for the USB 3.1 Specification. Figure 2-3 shows the partitioning described in this spec for the Converged IO 1.0 Specification.

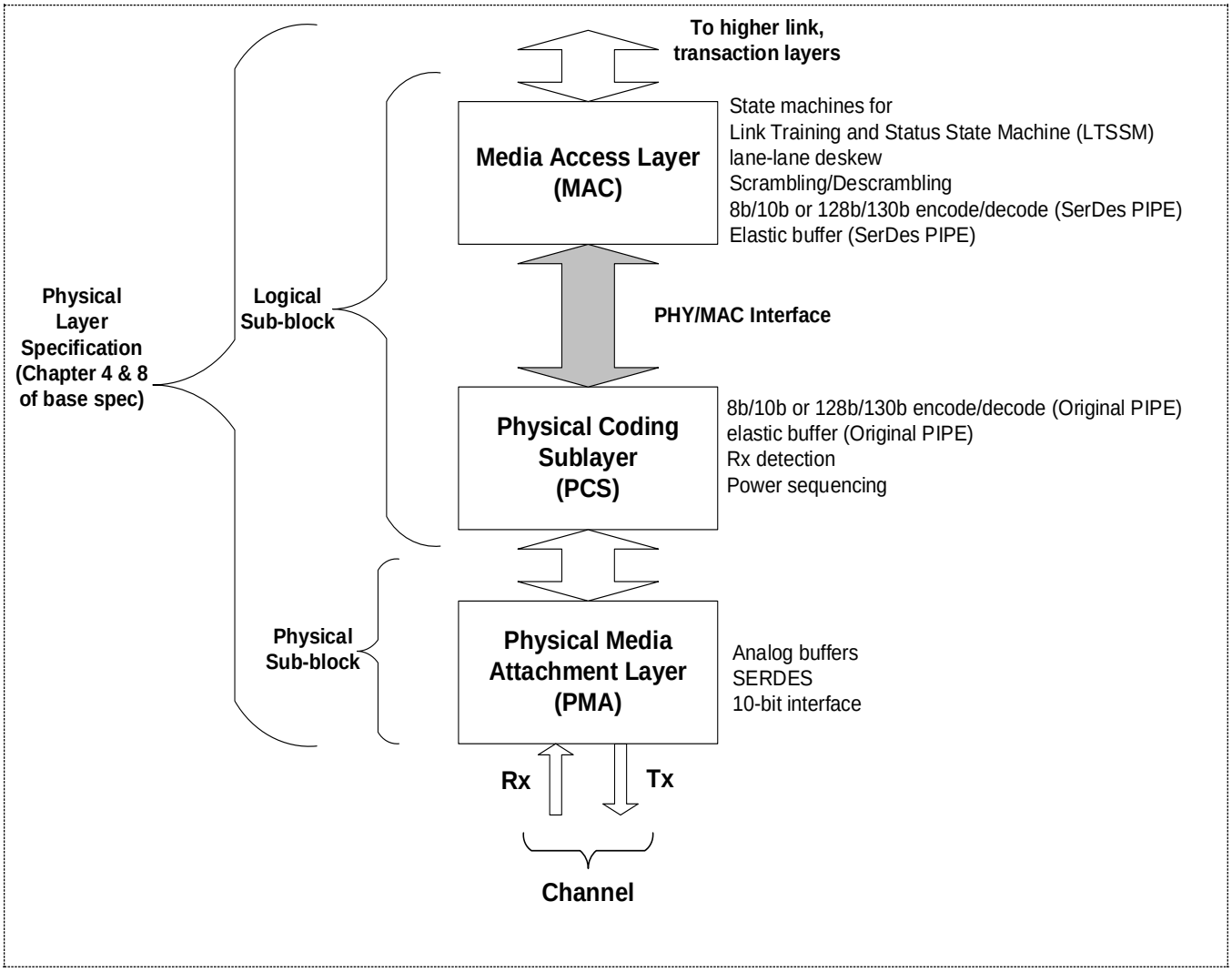


Figure 2-1: Partitioning PHY Layer for PCI Express

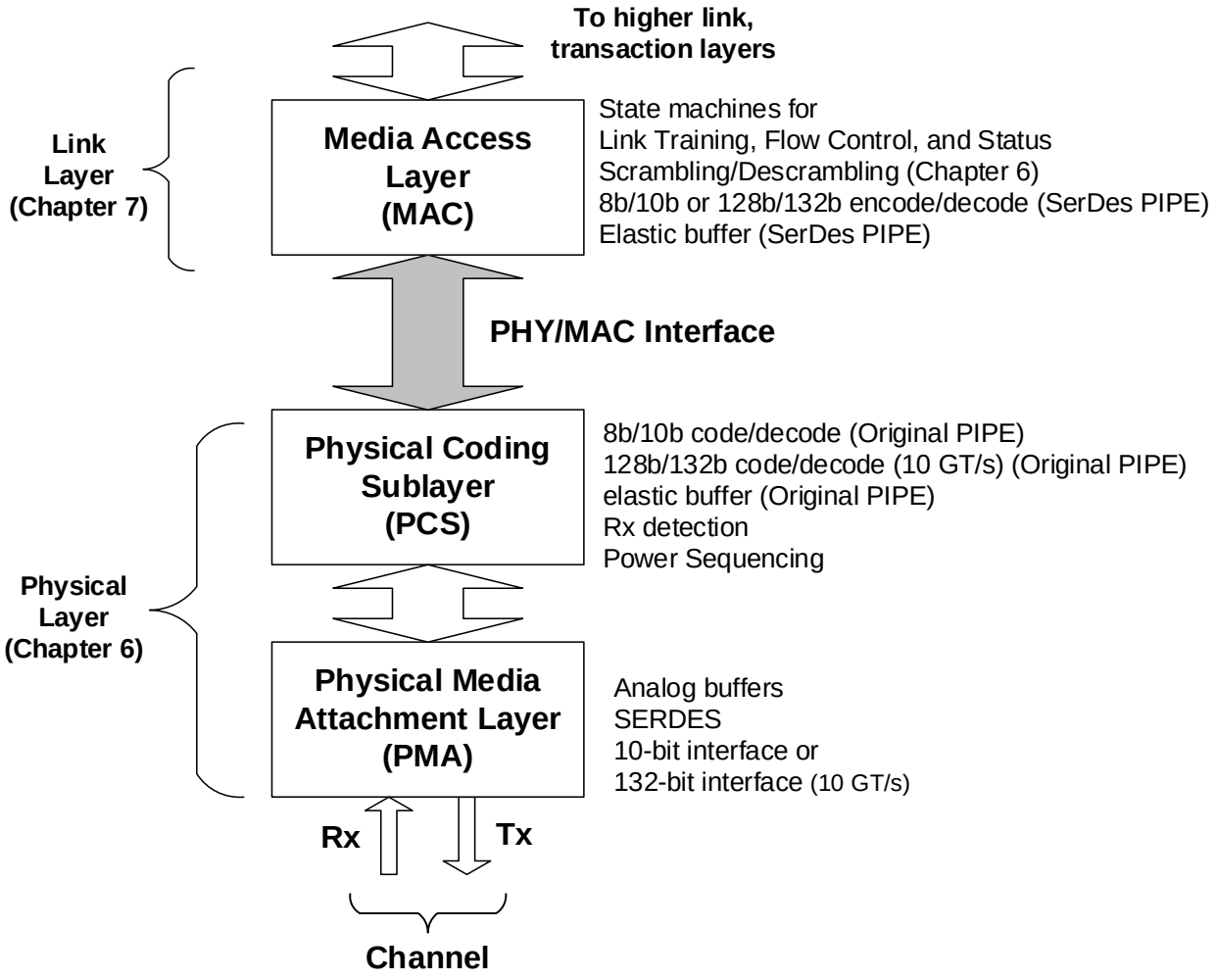


Figure 2-2 Partitioning PHY Layer for USB

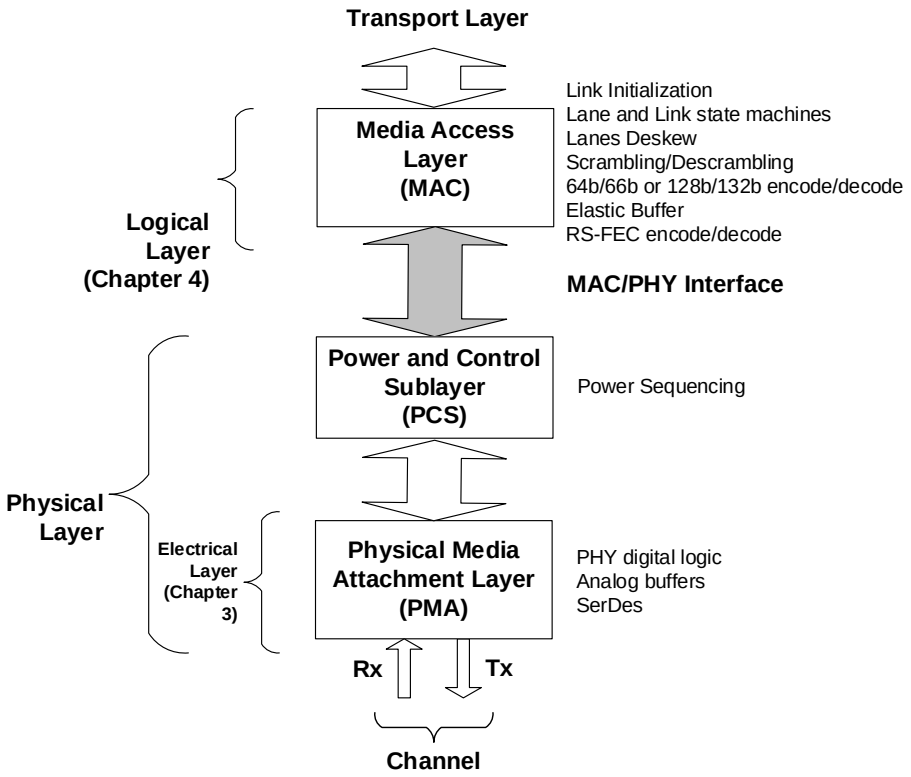


Figure 2-3. Partitioning PHY Layer for Converged IO

2.1 PCI Express PHY Layer

The PCI Express PHY Layer handles the low level PCI Express protocol and signaling. This includes features such as analog buffers, receiver detection, data serialization and de-serialization, 8b/10b encoding/decoding, 128b/130b encoding/decoding (8 GT/s), and elastic buffers. The primary focus of this block is to shift the clock domain of the data from the PCI Express rate to one that is compatible with the general logic in the ASIC.

Some key features of the PCI Express PHY are:

- Standard PHY interface enables multiple IP sources for PCI Express Logical Layer and provides a target interface for PCI Express PHY vendors.
- Supports 2.5GT/s only or 2.5GT/s and 5.0 GT/s, or 2.5 GT/s, 5.0 GT/s, and 8.0 GT/s, or 2.5 GT/s, 5.0 GT/s, 8.0 GT/s and 16 GT/s serial data transmission rate
- Utilizes 8-bit, 16-bit, 32 –bit, or 64-bit parallel interface to transmit and receive PCI Express data
- Allows integration of high speed components into a single functional block as seen by the endpoint device designer
- Data and clock recovery from serial stream on the PCI Express bus
- Holding registers to stage transmit and receive data
- Supports direct disparity control for use in transmitting compliance pattern(s)
- 8b/10b encode/decode and error indication
- 128b/130b encode/decode and error indication
- Receiver detection
- Beacon transmission and reception

- Selectable Tx Margining, Tx De-emphasis and signal swing values
- Lane Margining at the Receiver
- Polarity
- Electrical Idle Entry/Exit Detection (Squelch)

2.2 USB PHY Layer

The USB PHY Layer handles the low level USB protocol and signaling. This includes features such as analog buffers, receiver detection, data serialization and de-serialization, 8b/10b encoding/decoding, 128b/132b encoding/decoding (10 GT/s), and elastic buffers. The primary focus of this block is to shift the clock domain of the data from the USB rate to one that is compatible with the general logic in the ASIC.

Some key features of the USB PHY are:

- Standard PHY interface enables multiple IP sources for USB Link Layer and provides a target interface for USB PHY vendors.
- Supports 5.0 GT/s and/or 10 GT/s serial data transmission rate
- Utilizes 8-bit, 16-bit or 32-bit parallel interface to transmit and receive USB data
- Allows integration of high speed components into a single functional block as seen by the device designer
- Data and clock recovery from serial stream on the USB bus
- Holding registers to stage transmit and receive data
- 8b/10b encode/decode and error indication
- 128b/132b encode/decode and error indication
- Receiver detection
- Low Frequency Periodic Signaling (LFPS)

2.3 Converged IO PHY Layer

The Converged IO PHY Layer handles the low level Converged IO protocol and signaling. This includes features such as data serialization and de-serialization, analog buffers, and receiver detection.

Some key features of the Converged IO PHY:

- Standard PHY interface enables multiple IP sources for Converged IO Link Layer and provides a target interface for Converged IO PHY vendors.
- Supports 10 GT/s and/or 20 GT/s serial data transmission rate
- Utilizes 32-bit parallel interface to transmit and receive converged IO data
- Data and clock recovery from serial stream on the Converged IO bus
- Holding registers to stage transmit and receive data
- Low Frequency Periodic Signaling (LFPS)

2.4 SATA PHY Layer

The SATA PHY Layer handles the low level SATA protocol and signaling. This includes features such as analog buffers, data serialization and deserialization, 8b/10b encoding/decoding, and elastic buffers. The primary focus of this block is to shift the clock domain of the data from the SATA rate to one that is compatible with the general logic in the ASIC.

Some key features of the SATA PHY are:

- Standard PHY interface enables multiple IP sources for SATA controllers and provides a target interface for SATA PHY vendors.
- Supports 1.5 GT/s only or 1.5 GT/s and 3.0 GT/s, or 1.5 GT/s, 3.0 GT/s and 6.0 GT/s serial data transmission rate
- Utilizes 8-bit, 16-bit, or 32-bit parallel interface to transmit and receive SATA data
- Allows integration of high speed components into a single functional block as seen by the device designer
- Data and clock recovery from serial stream on the SATA bus
- Holding registers to stage transmit and receive data
- 8b/10b encode/decode and error indication
- COMINIT and COMRESET transmission and reception

2.5 Low Pin Count Interface and SerDes Architecture

To address the issue of increasing signal count, the message bus interface was introduced in PIPE 4.4 and utilized for PCIe lane margining at the receiver and elastic buffer depth control. In PIPE 5.0, all legacy PIPE signals without critical timing requirements were mapped into message bus registers so that their associated functionality could be accessed via the message bus interface instead of implementing dedicated signals. Any new features added in PIPE 4.4 and onwards are available only via message bus accesses unless they have critical timing requirements that need dedicated signals.

To facilitate the design of general purpose PHYs delivered as hard IPs and to provide the MAC with more freedom to do latency optimizations, a SerDes architecture was defined in PIPE 5.0. This architecture simplifies the PHY and shifts much of the protocol specific logic into the MAC.

To maximize interoperability between MAC and PHY IPs, PHY designs must adhere to the requirements stated in Table 2-1 for support of the legacy pin interface versus the low pin count interface and for support of the original PIPE architecture versus the SerDes architecture.

The legacy pin interface refers to a pin interface that utilizes all the applicable dedicated signals as well as the message bus interface for features not supported through dedicated signals. The low pin count interface refers to a pin interface that utilizes the message bus interface for all features supported through the message bus, using dedicated signals only for features not supported through the message bus. The legacy pin interface dedicated signals are defined in PIPE 4.4.1 and earlier and have been deprecated in PIPE 5.0.

The original PIPE architecture is represented in Figure 4-2, Figure 4-3, Figure 4-4 and Figure 4-5. The SerDes Architecture is represented in Figure 4-7 and Figure 4-8.

Table 2-1. PHY Requirements for Legacy Pin Interface vs Low Pin Count Interface and Original PIPE vs SerDes Architecture Support

	Converged IO/ DisplayPort	**USB 3.2 and Less**	**PCIe 5.0[1]**	**PCIe 4.0 and Less**	**SATA**
Legacy Pin Interface	Not allowed	Required (see version 4.4.1)	Not Allowed	Required (see version 4.4.1)	Required(see version 4.4.1)
Low Pin Count	Required	Optional	Required	Required for Gen4 RX	Optional

Interface				margining only, optional for everything else	
Original PIPE Architecture	Not allowed	Required	Recommended[2]	Required	Required
SerDes Architecture	Required	Optional	Required	Optional	Optional

1. PIPE support for PCIe 5.0 will be added into the next revision of the specification; this column is intended to indicate a direction for the PIPE architecture for PCIe 5.0.
2. To provide interoperability with PCIe and USB MACs that choose not to migrate to the SerDes architecture, PHYs are encouraged to provide support for original PIPE via a method where the associated logic can be easily optimized out. With this, designs that do not require a PHY that supports original PIPE are not burdened with any unneeded logic.

3 PHY/MAC Interface

Figure 3-1 shows the data and logical command/status signals between the PHY and the MAC layer. Figure 3-2 and Figure 3-3 shows the data and command/status signals between the PHY and the MAC layer for DisplayPort DPTX and DPRX, respectively. Full support of PCI Express mode, USB mode, Sata mode, DisplayPort mode, and Converged IO mode at all rates require different numbers of control and status signals to be implemented. Refer to Section 6.1 for details on which specific signals are required for each operating mode.

Figure 3-1. PHY/MAC Interface

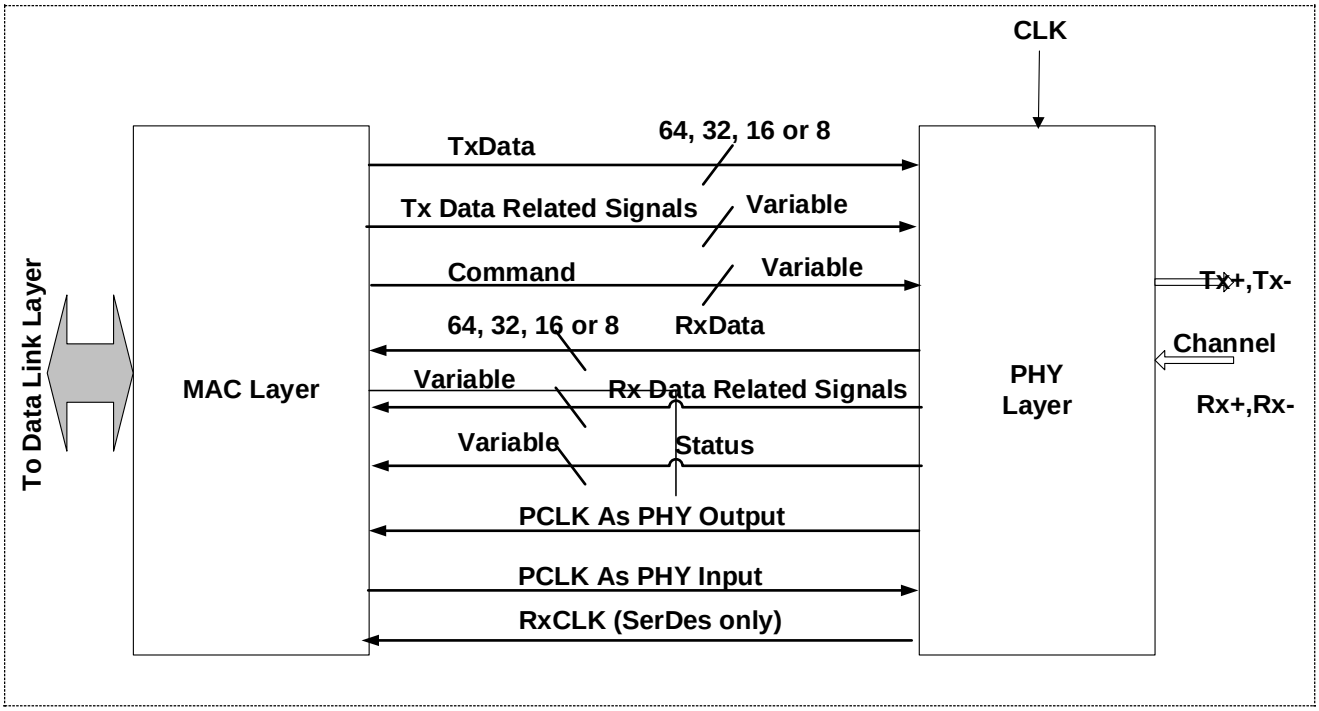


Figure 3-2. DPTX PHY/MAC Interface

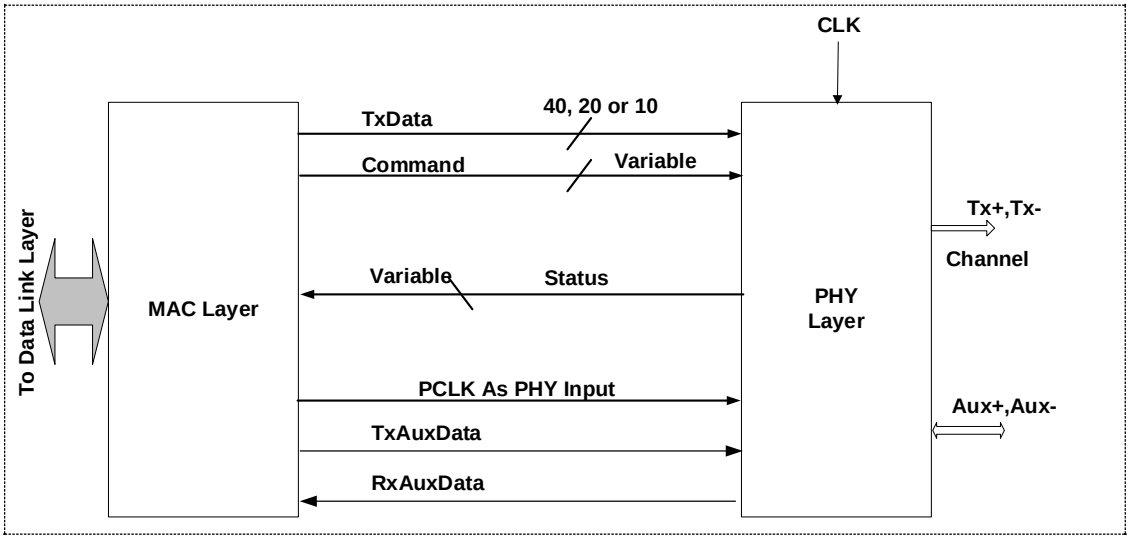
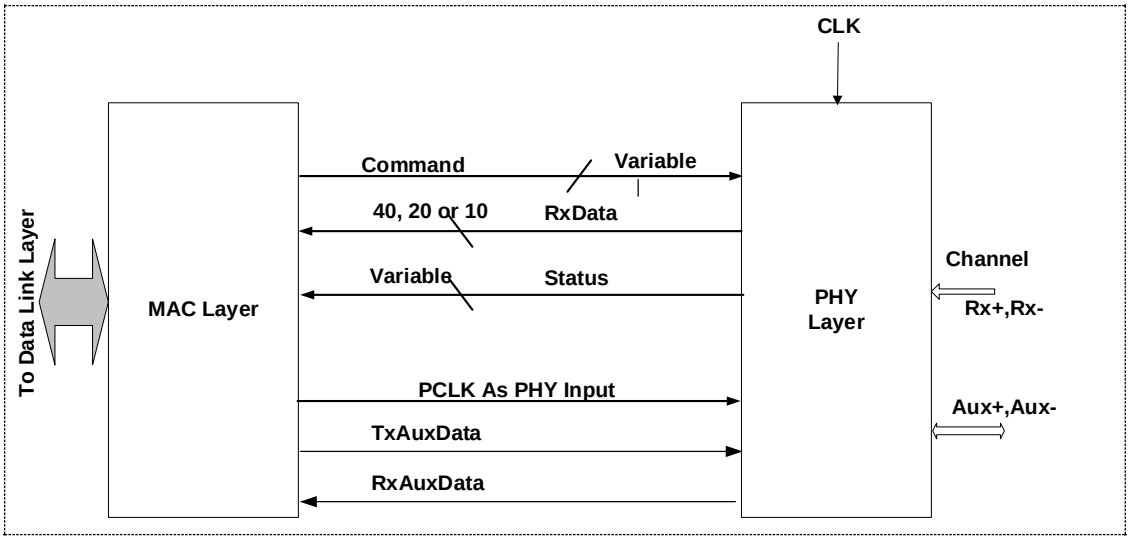


Figure 3-3. DPRX PHY/MAC Interface



This specification allows several different PHY/MAC interface configurations to support various signaling rates.

For PIPE implementations that support only the 2.5 GT/s signaling rate in PCI Express mode implementers can choose to have 16 bit data paths with PCLK running at 125 MHz, or 8 bit data paths with PCLK running at 250 MHz. PIPE implementations that support 5.0 GT/s signaling and 2.5 GT/s signaling in PCI Express mode, and therefore are able to switch between 2.5 GT/s and 5.0 GT/s signaling rates, can be implemented in several ways. An implementation may choose to have PCLK fixed at 250 MHz and use 8-bit data paths when operating at 2.5 GT/s signaling rate, and 16-bit data paths when operating at 5.0 GT/s signaling rate. Another implementation choice is to use a fixed data path width and change PCLK frequency to adjust the signaling rate. In this case, an implementation with 8-bit data paths would provide PCLK at 250 MHz for 2.5 GT/s signaling and provide PCLK at 500 MHz for 5.0 GT/s signaling. Similarly, an implementation with 16-bit data paths would provide PCLK at 125 MHz for 2.5 GT/s signaling and 250 MHz for 5.0 GT/s signaling. The full list of possibilities is shown in Table 3-1.

For PIPE implementations that support 5.0 GT/s USB mode and/or 10 GT/s USB mode implementers can choose from options shown in Table 3-2. A PIPE compliant MAC or PHY is only required to support one option for each USB transfer speed that it supports.

For SATA PIPE implementations that support only the 1.5 GT/s signaling rate implementers can choose to have 16 bit data paths with PCLK running at 75 MHz, or 8 bit data paths with PCLK running at 150, 300 or 600 MHz. The 300 and 600 Mhz options requires the use of TxDataValid and RxDataValid signals to toggle the use of data on the data bus.

SATA PIPE implementations that support 1.5 GT/s signaling and 3.0 GT/s signaling in SATA mode, and therefore are able to switch between 1.5 GT/s and 3.0 GT/s signaling rates, can be implemented in several ways. An implementation may choose to have PCLK fixed at 150 MHz and use 8-bit data paths when operating at 1.5 GT/s signaling rate, and 16-bit data paths when operating at 3.0 GT/s signaling rate. Another implementation choice is to use a fixed data path width and change PCLK frequency to adjust the signaling rate. In this case, an implementation with 8-bit data paths could provide PCLK at 150 MHz for 1.5 GT/s signaling and provide PCLK at 300 MHz for 3.0 GT/s signaling. Similarly, an implementation with 16-bit data paths would provide PCLK at 75 MHz for 1.5 GT/s signaling and 150 MHz for 3.0 mode are shown GT/s

signaling. The full set of possible widths and PCLK rates for SATA in Table 3-3. A PIPE compliant MAC or PHY is only required to support one option for each SATA transfer speed that it supports.

The full set of possible widths and PCLK rates for PCI Express mode is shown in Table 3-1. A PIPE compliant MAC or PHY is only required to support one option for each PCI Express transfer speed that it supports. Note that PHYs that support greater than x4 link widths must provide an option for 32-bit or less data width.

Table 3-1. PCI Express Mode - Possible PCLK rates and data widths

Mode	PCLK	Data Width	TxDataValid and RxDataValid Strobe Rate
2.5 GT/s	2000 Mhz	8 bits	1 in 8 PCLKs
2.5 GT/s	1000 Mhz	8 bits	1 in 4 PCLKs
2.5 GT/s	500 Mhz	8 bits	1 in 2 PCLKs
2.5 GT/s	250 Mhz	8 bits	N/A
2.5 GT/s	250 Mhz	16 bits	1 in 2 PCLKs
2.5 GT/s	500 Mhz	16 bits	1 in 4 PCLKs
2.5 GT/s	125 Mhz	16 bits	N/A
2.5 GT/s	250 Mhz	32 bits	1 in 4 PCLKs
2.5 GT/s	62.5 Mhz	32 bits	N/A
2.5 GT/s	62.5 Mhz	64 bits	1 in 2 PCLKs
2.5 GT/s	31.25 Mhz	64 bits	N/A
5.0 GT/s	2000 Mhz	8 bits	1 in 4 PCLKs
5.0 GT/s	1000 Mhz	8 bits	1 in 2 PCLKs
5.0 GT/s	500 Mhz	8 bits	N/A
5.0 GT/s	500 Mhz	16 bits	1 in 2 PCLKs
5.0 GT/s	250 Mhz	16 bits	N/A
5.0 GT/s	250 Mhz	32 bits	1 in 2 PCLKs
5.0 GT/s	125 Mhz	32 bits	N/A
5.0 GT/s	125 Mhz	64 bits	1 in 2 PCLKs

5.0 GT/s	62.5 Mhz	64 bits	N/A
8.0 GT/s	2000 Mhz	8 bits	1 in 2 PCLKs
8.0 GT/s	1000 Mhz	8 bits	N/A
8.0 GT/s	1000 Mhz	16 bits	1 in 2 PCLKs
8.0 GT/s	500 Mhz	16 bits	N/A
8.0 GT/s	500 Mhz	32 bits	1 in 2 PCLKs
8.0 GT/s	250 Mhz	32 bits	N/A
8.0 GT/s	250 Mhz	64 bits	1 in 2 PCLKs
8.0 GT/s	125 Mhz	64 bits	N/A
16.0 GT/s	2000 Mhz	8 bits	N/A
16.0 GT/s	1000 Mhz	16 bits	N/A
16.0 GT/s	500 Mhz	32 bits	N/A
16.0 GT/s	250 Mhz	64 bits	N/A

Figure 3-4. PCI Express Mode (SerDes only) -- Possible RxCLK Rates and Data Widths

Mode	RxCLK	Data Width
2.5 GT/s	250 Mhz	8 bits
2.5 GT/s	125 Mhz	16 bits
2.5 GT/s	62.5 Mhz	32 bits
2.5 GT/s	31.25 Mhz	64 bits
5.0 GT/s	500 Mhz	8 bits
5.0 GT/s	250 Mhz	16 bits
5.0 GT/s	125 Mhz	32 bits
5.0 GT/s	62.5 Mhz	64 bits
8.0 GT/s	1000 Mhz	8 bits
8.0 GT/s	500 Mhz	16 bits

8.0 GT/s	250 Mhz	32 bits
8.0 GT/s	125 Mhz	64 bits
16.0 GT/s	2000 Mhz	8 bits
16.0 GT/s	1000 Mhz	16 bits
16.0 GT/s	500 Mhz	32 bits
16.0 GT/s	250 Mhz	64 bits

Table 3-2. USB Mode – Possible PCLK or RxClk rates and data widths

Mode	PCLK or RxClk	Data Width
5.0 GT/s USB	125 Mhz	32 bits
5.0 GT/s USB	250 Mhz	16 bits
5.0 GT/s USB	500 Mhz	8 bits
10.0 GT/s USB	312.5 Mhz	32 bits
10.0 GT/s USB	625 Mhz	16 bits
10.0 GT/s USB	1250 Mhz	8 bits

Table 3-3. SATA Mode – Possible PCLK rates and data widths

Mode	PCLK	Data Width	TxDataValid/RxDataValid Strobe Rate
1.5 GT/s SATA	600 Mhz	8 bits	1 in 4 PCLKs
1.5 GT/s SATA	300 Mhz	8 bits	1 in 2 PCLKs
1.5 GT/s SATA	150 Mhz	8 bits	N/A
1.5 GT/s SATA	75 Mhz	16 bits	N/A
1.5 GT/s SATA	37.5 Mhz	32 bits	N/A
3.0 GT/s SATA	300 Mhz	8 bits	N/A
3.0 GT/s SATA	150 Mhz	16 bits	N/A
3.0 GT/s SATA	75 Mhz	32 bits	N/A
3.0 GT/s SATA	600 Mhz	8 bits	1 in 2 PCLKs
6.0 GT/s SATA	600 Mhz	8 bits	N/A
6.0 GT/s SATA	300 Mhz	16 bits	N/A
6.0 GT/s SATA	150 Mhz	32 bits	N/A

Note: In SATA Mode if the PHY elasticity buffer is operating in nominal empty mode – then

RxDataValid may also be used when the EB is empty and no data is available.

Figure 3-5. SATA Mode (SerDes only) – Possible RxCLK Rates and Data Widths

Mode	RxCLK	Data Width
1.5 GT/s SATA	150 Mhz	8 bits
1.5 GT/s SATA	75 Mhz	16 bits
1.5 GT/s SATA	37.5 Mhz	32 bits
3.0 GT/s SATA	300 Mhz	8 bits
3.0 GT/s SATA	150 Mhz	16 bits
3.0 GT/s SATA	75 Mhz	32 bits
6.0 GT/s SATA	600 Mhz	8 bits
6.0 GT/s SATA	300 Mhz	16 bits
6.0 GT/s SATA	150 Mhz	32 bits

Table 3-4 shows possible PCLK and data width options for DisplayPort implementations.

Table 3-4 DPTX and DPRX Mode – Possible PCLK or RxCLK Rates and Data Widths

Mode	PCLK/RxCLK	Data Width
1.62 Gbps DisplayPort	162 Mhz	10 bits
	81 Mhz	20 bits
	40.5 Mhz	40 bits
2.7 Gbps DisplayPort	270 Mhz	10 bits
	135 Mhz	20 bits
	62.52 Mhz	40 bits
5.4 Gbps DisplayPort	540 Mhz	10 bits
	270 Mhz	20 bits
	135 Mhz	40 bits
8.1 Gbps DisplayPort	810 Mhz	10 bits
	405 Mhz	20 bits
	202.5 Mhz	40 bits

Note: When a MAC that implements the TxDataValid signal is using a mode that does not use TxDataValid the MAC shall keep TxDataValid asserted. When a PHY that implements RxDataValid is in a mode that does not use RxDataValid the PHY shall keep RxDataValid

asserted.

There may be PIPE implementations that support multiples of the above configurations. PHY implementations that support multiple configurations at the same rate must support the width and PCLK rate control signals. A PHY that supports multiple rates in PCI Express Mode or SATA Mode or USB Mode must support configurations across all supported rates that are fixed PCLK rate. A PHY that supports multiple rates in PCI Express Mode or SATA Mode must support configurations across all supported rates that are fixed data path width.

4 PCI Express, USB, and Converged IO PHY Functionality

Figure 4-1 shows the functional block diagram of the PHY. The functional blocks shown are not intended to define the internal architecture or design of a compliant PHY but to serve as an aid for signal grouping.

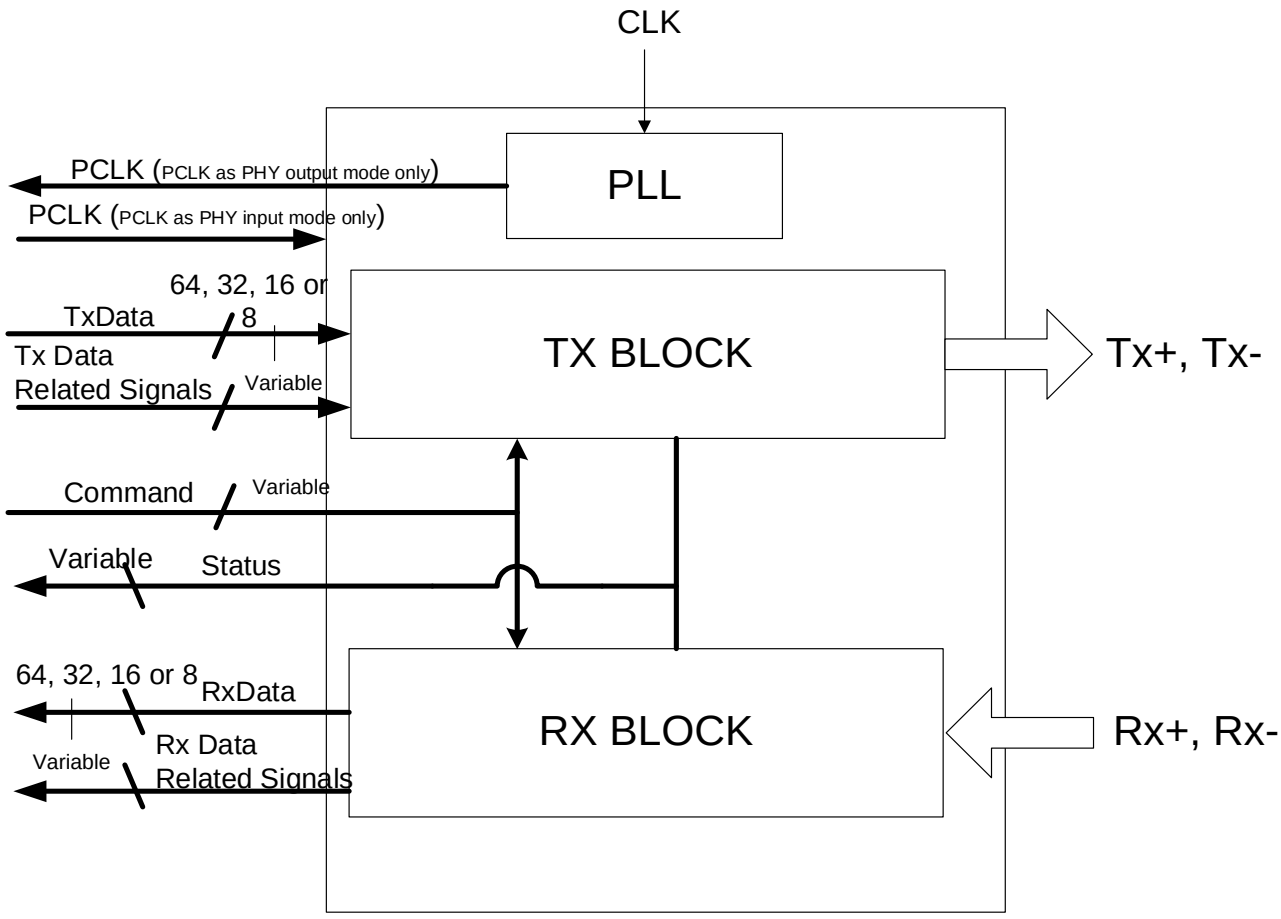


Figure 4-1: PHY Functional Block Diagram

Sections below provide descriptions of each of the blocks shown in Figure 4-1: PHY Functional Block Diagram. These blocks represent high-level functionality that is required to exist in the PHY implementation. These descriptions and diagrams describe general architecture and behavioral characteristics. Different implementations are possible and acceptable.

4.1 Original PIPE Architecture

4.1.1 Transmitter Block Diagram (2.5 and 5.0 GT/s)

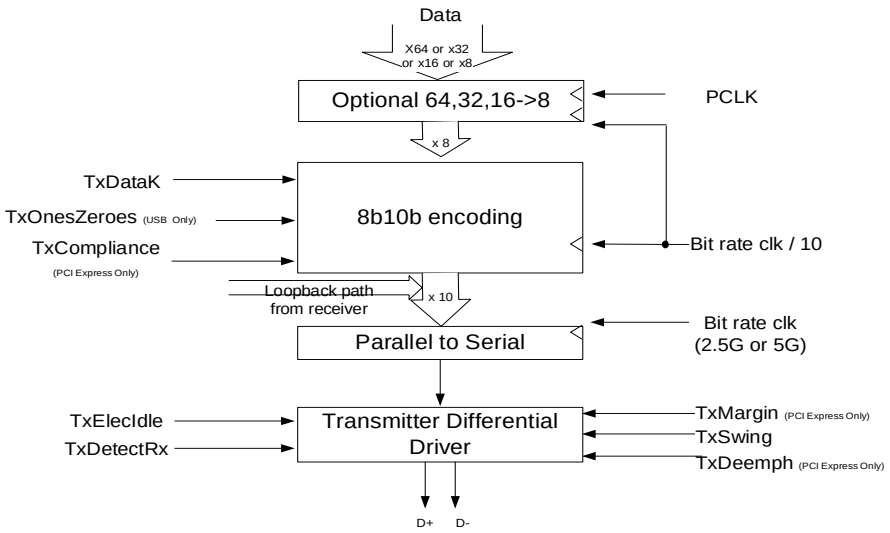


Figure 4-2: Transmitter Block Diagram

4.1.2 Transmitter Block Diagram (8.0/10/16 GT/s)

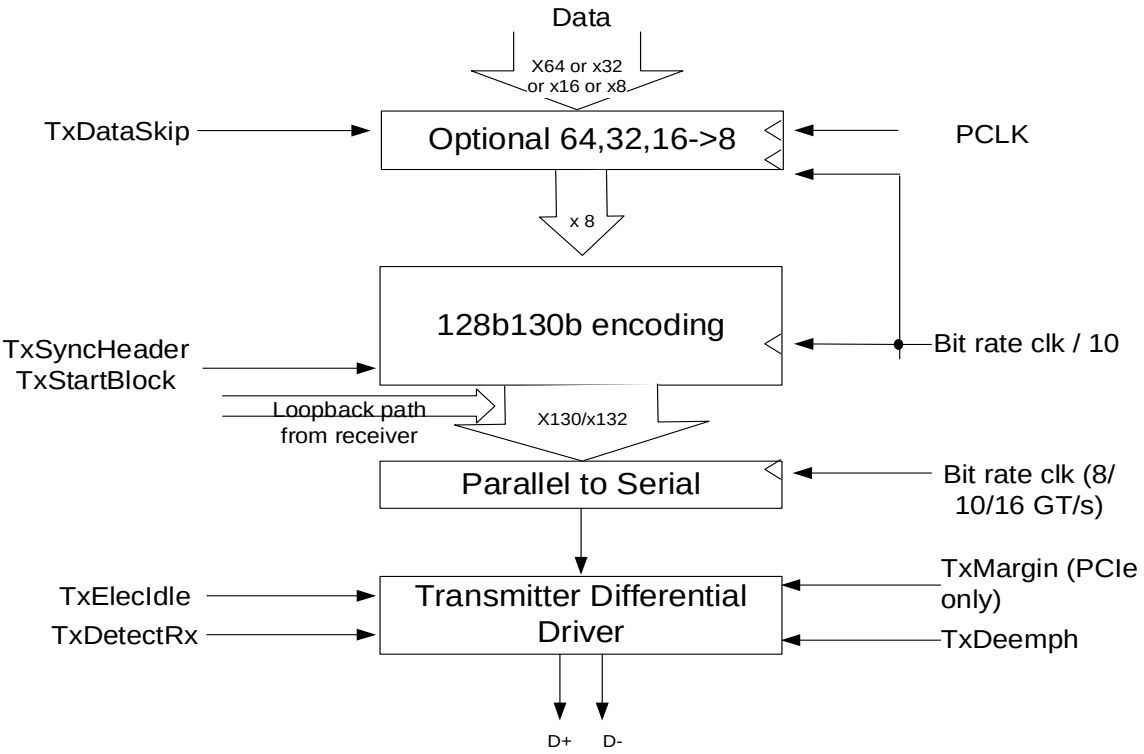


Figure 4-3: Transmitter Block Diagram (8.0/10/16 GT/s)

4.1.3 Receiver Block Diagram (2.5 and 5.0 GT/s)

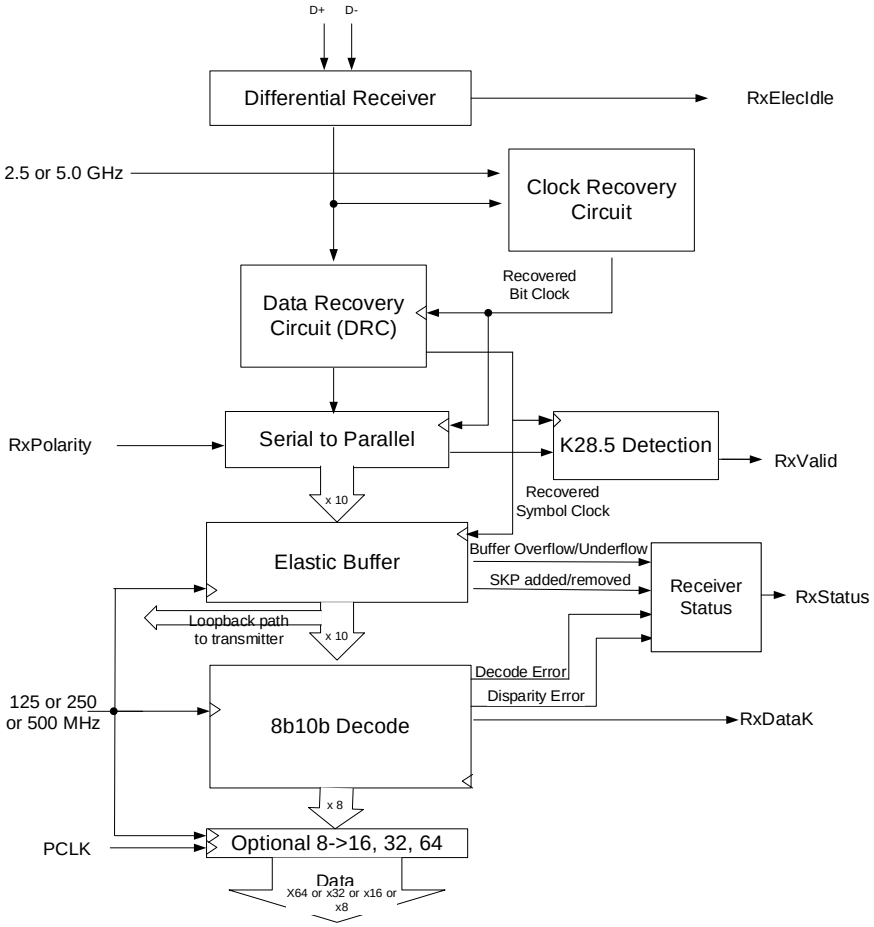


Figure 4-4: Receiver Block Diagram

4.1.4 Receiver Block Diagram (8.0/10.0/16 GT/s)

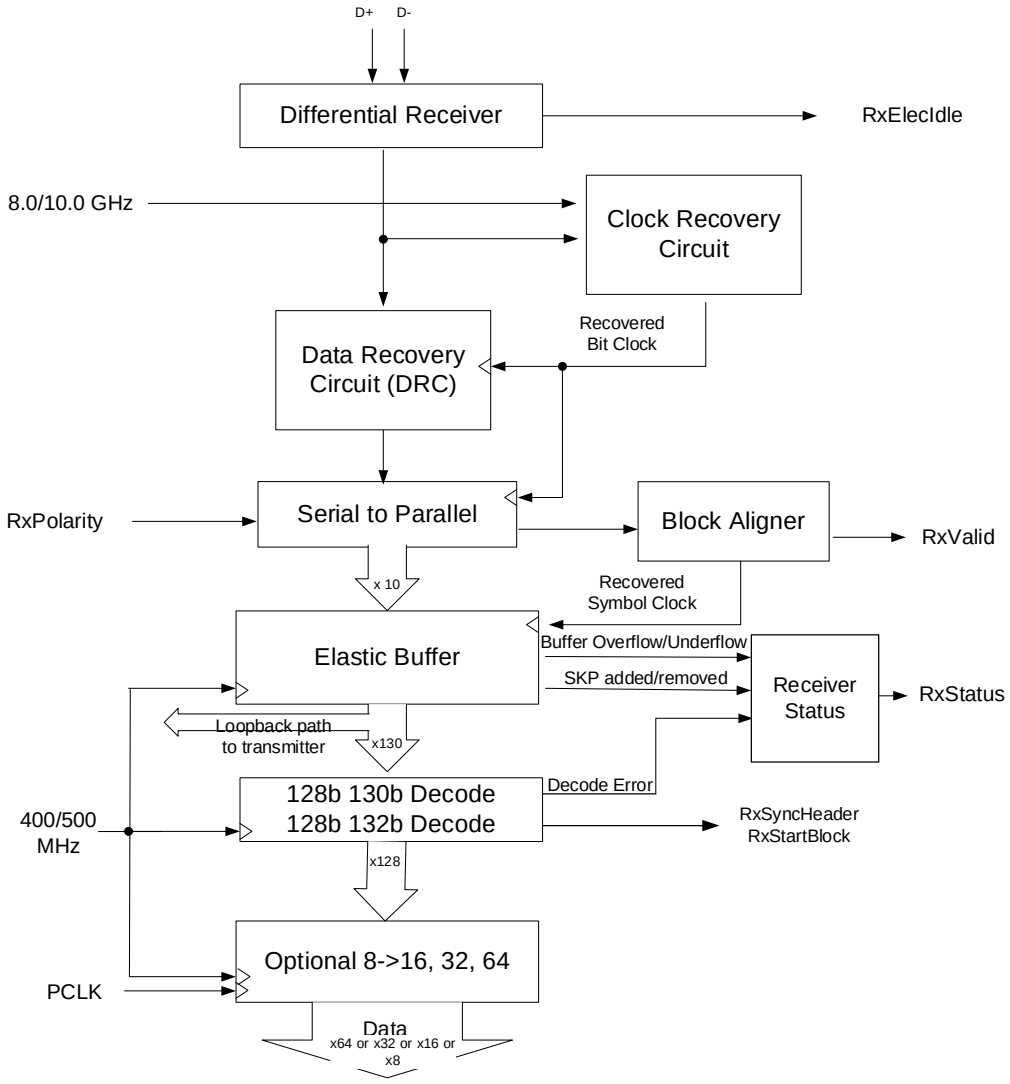


Figure 4-5: Receiver Block Diagram (8.0/10/16 GT/s)

4.1.5 Clocking

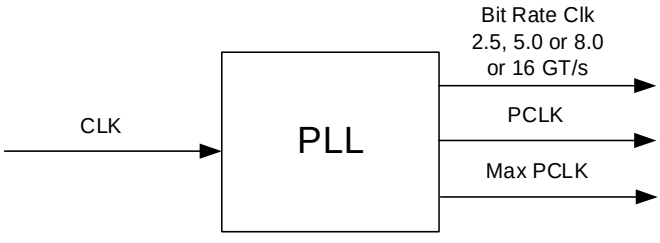


Figure 4-6: Clocking Block Diagram

4.2 SerDes Architecture

With the SerDes architecture, the PHY implements minimal digital logic compared to the original PIPE architecture. Figure 4-7 shows the transmitter functionality implemented in the PHY. The data received from the MAC goes through a parallel to serial converter before being driven out on differential wires. Note that in the SerDes architecture, all loopback logic resides in the MAC. Figure 4-8 shows the receiver functionality implemented in the PHY. The data received on the input differential wires goes through a serial to parallel converter before being forwarded to the MAC along with a recovered clock, RxCLK.

4.2.1 SerDes Architecture: Transmitter Block Diagram

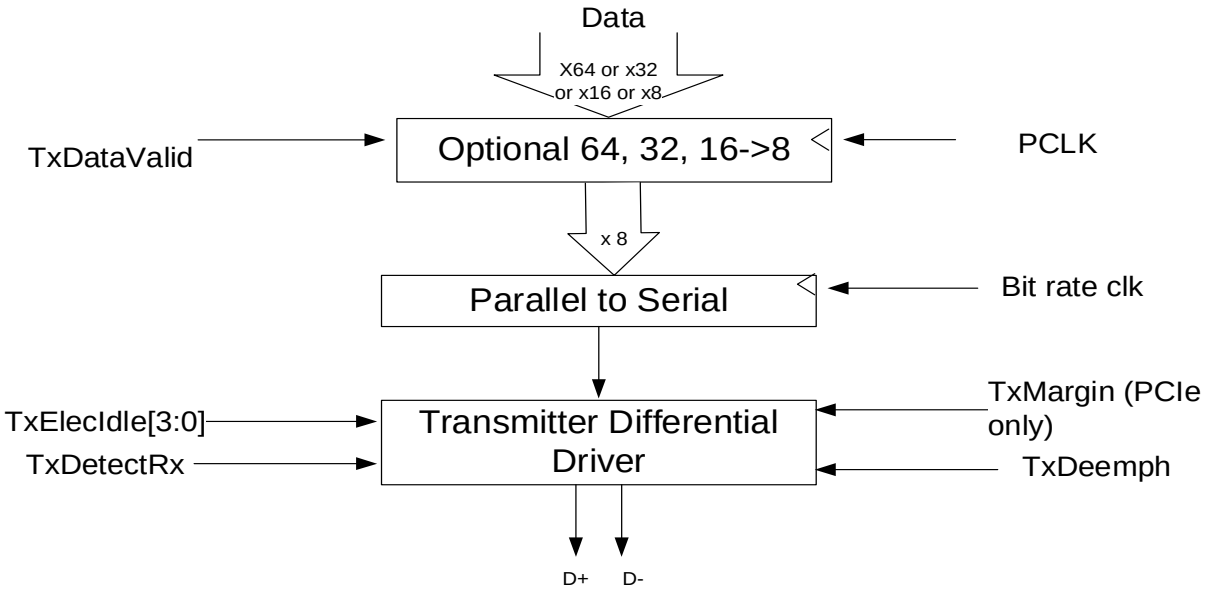


Figure 4-7. SerDes Architecture: PHY Transmitter Block Diagram

4.2.2 SerDes Architecture: Receiver Block Diagram

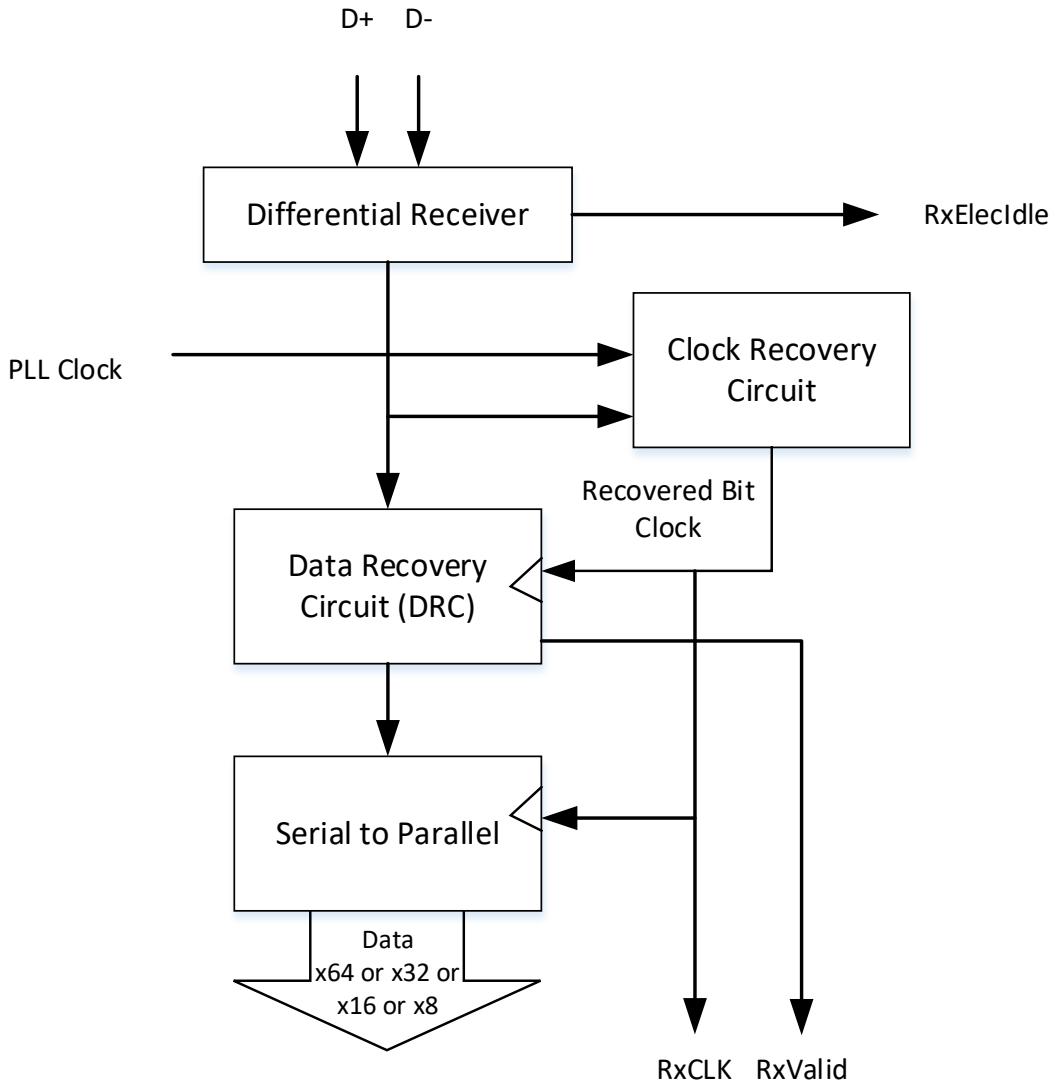


Figure 4-8. SerDes Architecture: PHY Receiver Block Diagram

5 SATA PHY Functionality

Figure 4-1 shows the functional block diagram of a SATA PHY. The functional blocks shown are not intended to define the internal architecture or design of a compliant PHY but to serve as an aid for signal grouping.

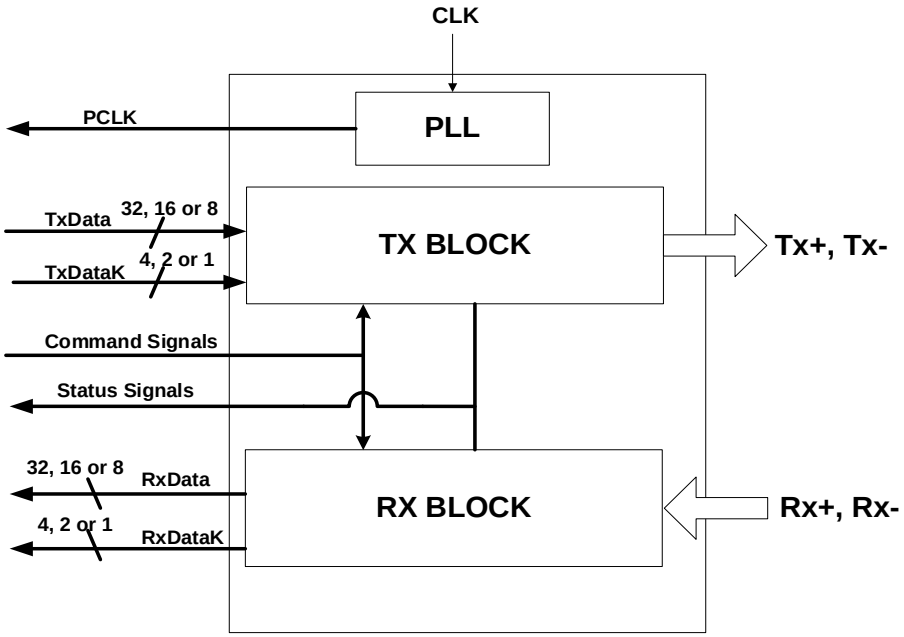


Figure 5-1: PHY Functional Block Diagram

Sections below provide descriptions of each of the blocks shown in Figure 5-1. These blocks represent high-level functionality that is required to exist in the PHY implementation. These descriptions and diagrams describe general architecture and behavioral characteristics. Different implementations are possible and acceptable.

5.1 Transmitter Block Diagram (1.5, 3.0, and 6.0 GT/s)

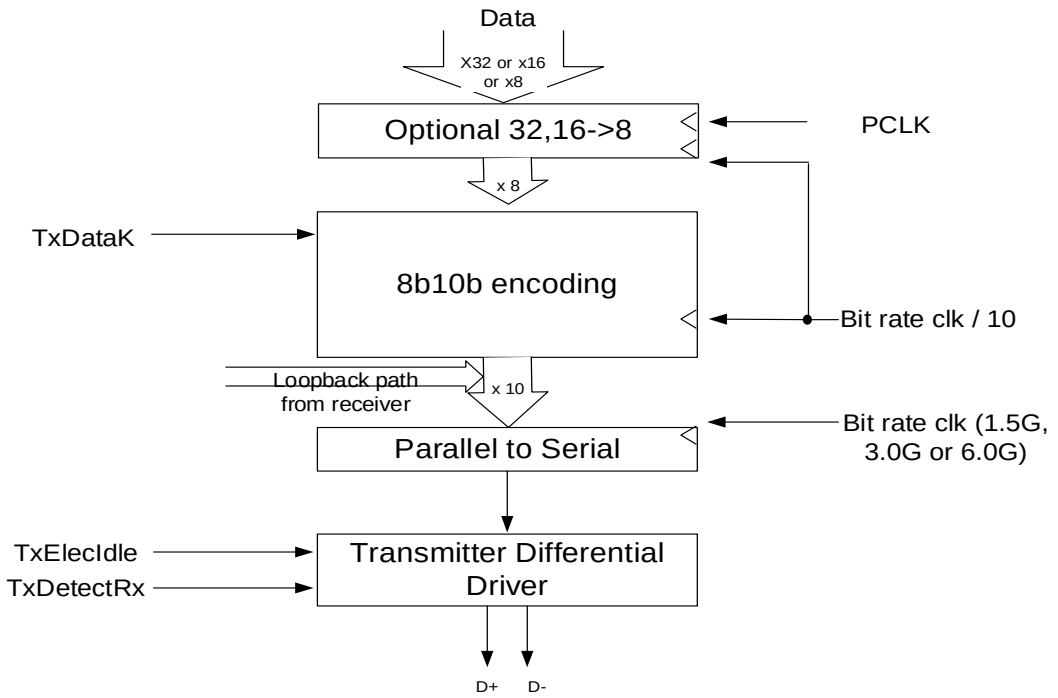


Figure 5-2: Transmitter Block Diagram (1.5, 3.0, and 6.0 GT/s)

5.2 Receiver Block Diagram (1.5, 3.0 and 6.0 GT/s)

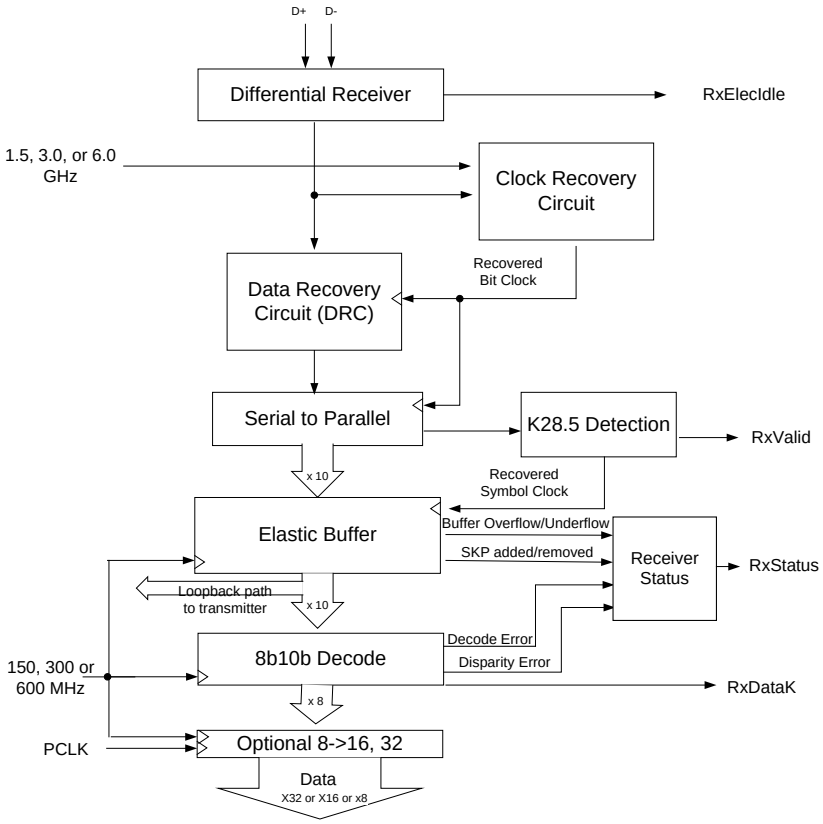


Figure 5-3: Receiver Block Diagram (1.5, 3.0 and 6.0 GT/s)

5.3 Clocking

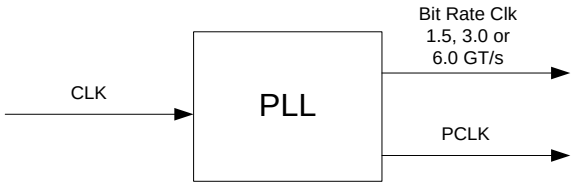


Figure 5-4: Clocking Block Diagram

6 PIPE Interface Signal Descriptions

The PHY input and output signals are described in the following tables. Note that Input/Output is defined from the perspective of a PIPE compliant PHY component. Thus a signal described as an "Output" is driven by the PHY and a signal described as an "Input" is received by the PHY. A basic description of each signal is provided. More details on their operation and timing can be found in following sections. All signals on the 'parallel' side of a PIPE implementation are synchronous with PCLK, with exceptions noted in the tables below. PHYs that only support SerDes architecture do not require the signals marked as "not used in the SerDes architecture"; however, PHYs that support both original PIPE and SerDes architecture must implement all the signals.

Notes: For Converged IO and DisplayPort, the low speed side channel is not part of the PIPE definition; however the appendix does list DisplayPort AUX signals.

6.1 PHY/MAC Interface Signals – Common for SerDes and Original PIPE

This section describes signals that are applicable to both SerDes architecture and Original PIPE. Any deltas in usage between the two architectures are noted in the description.

6.1.1 Data Interface

Table 6-1. Transmit Data Interface Input Signals

Name	Active Level	Description	Relevant Protocols
TxData[63:0] for 64-bit interface TxData[31:0] for 32-bit interface TxData[15:0] for 16-bit interface TxData[7:0] for 8-bit interface	N/A	Parallel data input bus. For the 16-bit interface, 16 bits represent 2 symbols of transmit data. Bits [7:0] are the first symbol to be transmitted, and bits [15:8] are the second symbol. For the 32-bit interface, 32 bits represent the 4 symbols of transmit data. Bits [23:16] are the third symbol to be transmitted, and bits [31:24] are the fourth symbol. For the 64-bit interface, 64 bits represent 8 symbols of transmit data. Bits [39:32], bits [47:40], bits [55:48] and bits [63:46] are the fifth, sixth, seventh, and eighth symbols, respectively. Bit zero is the first to be transmitted. For SerDes architecture, the TxData signal width options are 80, 40, 20, and 10 bits. For block encoded data, only 8 bits out of each 10-bit slice are used, e.g. [7:0] represent byte0, [9:8] are reserved, [17:10] represent byte1, and [19:18] are reserved.	PCIe, SATA, USB, DisplayPort TX, Converged IO
TxDataValid	N/A	PCI Express Mode and SATA Mode: This signal allows the MAC to instruct the PHY to ignore the data interface for one clock cycle. A value of one indicates the phy will use the data, a value of zero indicates the phy will not use the data. It is recommended that the MAC assert TxDataValid at all times when the PHY is in a mode that does not require the signal. All PCI Express modes at 8 GT/s and 16 GT/s and all USB modes at 10 GT/s use TxDataValid. Refer to **Table 3-1**, Table 3-2, and Table 3-3 for a list of other modes that use TxDataValid.	PCIe, SATA

Table 6-2. Transmit Data Interface Output Signals

Name	Active Level	Description	Relevant Protocols
Tx+, Tx-	N/A	The differential outputs from the PHY. All transmitters shall be AC coupled to the media. See section 4.3.1.2 of the PCI Express Base Specification or section 6.2.2 of the USB 3.1 Specification.	PCIe, SATA, USB, DisplayPort TX

Table 6-3. Receive Data Interface Input Signals

Name	Active Level	Description	Relevant Protocols
Rx+, Rx-	N/A	The differential inputs to the PHY.	PCIe, SATA, USB, DisplayPort RX

Table 6-4. Receive Data Interface Output Signals

Name	Active Level	Description	Relevant Protocols
RxData[63:0] for 64-bit interface RxData[31:0] for 32-bit interface RxData[15:0] for 16-bit interface or RxData[7:0] for 8-bit interface	N/A	Parallel data output bus. For 16-bit interface, 16 bits represents 2 symbols of receive data. Bits [7:0] are the first symbol received, and bits [15:8] are the second symbol. For the 32 bit interface, 32 bits represent the 4 symbols of receive data. Bits [23:16] are the third symbol received, and bits [31:24] are the fourth symbol received. For the 64-bit interface, 64 bits represent 8 symbols of transmit data. Bits [39:32], bits [47:40], bits [55:48] and bits [63:46] are the fifth, sixth, seventh, and eighth symbols, respectively. Bit zero is the first bit received. When the PHY is in a SATA mode, the first valid data following an ALIGN primitive must appear as byte 0 in the receive data. For SerDes architecture, the TxData signal width options are 80, 40, 20, and 10 bits. For block encoded data, only 8 bits out of each 10-bit slice are used, e.g. [7:0] represent byte0, [9:8] are reserved, [17:10] represent byte1, and [19:18] are reserved.	PCIe, SATA, USB, DisplayPort RX, Converged IO

6.1.2 Command Interface

Table 6-5. Command Interface Input Signals

<table>
<tr><th>Name</th><th>Active Level</th><th>Description</th><th>Relevant Protocols</th></tr>
<tr><td>PHY Mode[3:0]</td><td>N/A</td><td>Selects PHY operating mode.<table><tr><td>Value</td><td>Description</td></tr><tr><td>0</td><td>PCI Express</td></tr><tr><td>1</td><td>USB</td></tr><tr><td>2</td><td>SATA</td></tr><tr><td>3</td><td>DisplayPort</td></tr><tr><td>4</td><td>Reserved</td></tr><tr><td>5</td><td>Reserved</td></tr><tr><td>6</td><td>Reserved</td></tr><tr><td>7</td><td>Converged IO</td></tr><tr><td>All others</td><td>Reserved</td></tr></table>Implementation of this signal is not required for PHYs that only support only a single mode.</td><td>PCIe, SATA, USB, DisplayPort, Converged IO</td></tr>
<tr><td>DP_Mode_TX_RX</td><td>N/A</td><td>This signal is used to distinguish between DPTX and DPRX when PHY Mode=0x3. A value of ‘0’ specifies DPTX; a value of ‘1’ specificies DPRX.</td><td>DisplayPort</td></tr>
<tr><td>SerDesArch</td><td>N/A</td><td>This signal indicates whether SerDes architecture is enabled. Displayport and Converged IO must always set this to ‘1’.</td><td>PCIe, SATA, USB, DisplayPort, Converged IO</td></tr>
<tr><td>SRISEnable</td><td>High</td><td>Used to tell the PHY to configure itself to support SRIS for PCIe.

SRISEnable must be set by the MAC before the first receiver detection. The PHY internally does sequencing and gates the exit to P0 with having setup for SRIS if SRISEnable is asserted.

For PCLK as PHY output, this signal must be set before the PLL is configured.</td><td>PCIe</td></tr>
</table>

TxDetectRx/ Loopback	High	Used to tell the PHY to begin a receiver detection operation or to begin loopback or to signal LFPS during P0 for USB Polling state. Refer to Sections 8.22 and 8.23 for details on the required values for all control signals to perform loopback and receiver detection operations and to signal Polling.LFPS. For receive detect in PHY power states where PCLK can be gated, this signal is asynchronous; in all other states, it is synchronous to PCLK. Converged IO Mode: Used to tell the PHY to signal LFPS. Sata Mode: Loopback support is optional for SATA PHYs. Loopback is only valid in Sata Mode when EncodeDecodeBypass is asserted. The RX elasticity buffer must be active during loopback. If the PHY runs out of data to transmit during loopback – it must transmit ALIGNs. TxDetectRX is not used in SATA mode.	PCIe, SATA, USB

TxElecIdle[3:0]	High	Forces Tx output to electrical idle when asserted except in loopback. See Section 8.22 (PCI Express Mode) or Section 8.23 (USB mode and Converged IO Mode) or Section 8.24 (SATA Mode) for the full description and usage of this pin. Note: The MAC must always have TxDataValid asserted when TxElecIdle transitions to either asserted or deasserted; TxDataValid is a qualifier for TxElecIdle sampling. See section 8.3 for the definitions of PHY power states. For original PIPE architecture, only bit 0 of this signal is used and all other bits are reserved. For SerDes architecture in PCIe mode, one bit is required per two symbols of interface data. For example, for an eight symbol wide interface, bit 0 would apply to symbols 0 and 1, bit 1 would apply to symbols 2 and 3, bit 2 would apply to symbols 4 and 5, bit 3 would apply to symbols 6 and 7. For narrower interfaces, unused bits of this signal are reserved.	PCIe, SATA USB, Converged IO
Reset#	Low	Resets the transmitter and receiver. This signal is asynchronous. The PHY reports its default power state after reset as defined in section 8.2.	PCIe, SATA, USB, DisplayPort

<table>
<tr>
<td>PowerDown[3:0]</td>
<td>N/A</td>
<td>Power up or down the transceiver.
Power states
PCI Express Mode:
<table>
<tr><th>[3]</th><th>[2]</th><th>[1]</th><th>[0]</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>P0, normal operation</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>P0s, low recovery time latency, power saving state</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>P1, longer recovery time latency, lower power state</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>P2, lowest power state</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>POWER_STATE_4 Phy specific</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>POWER_STATE_5 Phy specific</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>POWER_STATE_6 Phy specific</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>POWER_STATE_7 Phy specific</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>POWER_STATE_8 Phy specific</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>POWER_STATE_9 Phy specific</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>POWER_STATE_10 Phy specific</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>POWER_STATE_11 Phy specific</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>POWER_STATE_12 Phy specific</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td><td>POWER_STATE_13 Phy specific</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td><td>POWER_STATE_14 Phy specific</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>POWER_STATE_15 Phy specific</td></tr>
</table>
In PCLK as PHY output mode, when transitioning from P2 to P1, the signaling is asynchronous (since PCLK is not running).

A PIPE phy that supports PCI Express L1 PM Substates managed exclusively via this PowerDown signal must support at least one PHY specific power state meeting each of the requirements shown in the following table. If the PHY supports multiple suitable states with different exit latencies it is the</td>
<td>PCIe, USB, Converged IO</td>
</tr>
</table>

responsibility of the Mac to decide which states to use.

PCLK State	TX Common Mode State	RxElecIdle Supported	Exit Latency to P0
Off	Off	No	Implementation Specific
Off	On	No	Implementation Specific
Off	On	Yes	Implementation Specific

When managing L1 substates via sideband signals, the PHY must define at least one PowerDown encoding where PCLK can be turned off and TxCommonModeState and RxElecIdle are controlled through TxCommonModeDisable and RxEIDetectDisable; in this case, the MAC must hold the PowerDown value constant when in L1 substates.

USB Mode and Converged IO Mode:

[3]	[2]	[1]	[0]	Description
0	0	0	0	P0, normal operation
0	0	0	1	P1, low recovery time latency, power saving state
0	0	1	0	P2, longer recovery time latency, lower power state
0	0	1	1	P3, lowest power state
0	1	0	0	**POWER_STATE_4 Phy specific**
0	1	0	1	**POWER_STATE_5 Phy specific**
0	1	1	0	**POWER_STATE_6 Phy specific**
0	1	1	1	**POWER_STATE_7 Phy specific**

		1	0	0	0	**POWER_STATE_8 Phy specific**
		1	0	0	1	**POWER_STATE_9 Phy specific**
		1	0	1	0	**POWER_STATE_10 Phy specific**
		1	0	1	1	**POWER_STATE_11 Phy specific**
		1	1	0	0	**POWER_STATE_12 Phy specific**
		1	1	0	1	**POWER_STATE_13 Phy specific**
		1	1	1	0	**POWER_STATE_14 Phy specific**
		1	1	1	1	**POWER_STATE_15 Phy specific**
		When transitioning from P3 to P0, the signaling is asynchronous (since PCLK is not running).				

<table>
<tr>
<td>PowerDown[2:0]
Sata Mode</td>
<td>N/A</td>
<td>Sata Mode:

Power up or down the transceiver.
Power states

<table>
<tr><th>[3]</th><th>[2]</th><th>[1]</th><th>[0]</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>POWER_STATE_0 Operational state</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>POWER_STATE_1 Phy specific</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>POWER_STATE_2 Phy specific</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>POWER_STATE_3 Phy specific</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>POWER_STATE_4 Phy specific</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>POWER_STATE_5 Phy specific</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>POWER_STATE_6 Phy specific</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>POWER_STATE_7 Phy specific</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>POWER_STATE_8 Phy specific</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>POWER_STATE_9 Phy specific</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>POWER_STATE_10 Phy specific</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>POWER_STATE_11 Phy specific</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>POWER_STATE_12 Phy specific</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td><td>POWER_STATE_13 Phy specific</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td><td>POWER_STATE_14 Phy specific</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>POWER_STATE_15 Phy specific</td></tr>
</table>

A PIPE compliant SATA PHY is recommended to support at least 4 states other than POWER_STATE_0. There must be at least one power state meeting each of the requirements shown in the following table

<table>
<tr><th>PCLK State</th><th>TX Common Mode State</th><th>Exit La POWE _0</th></tr>
<tr><td>Off</td><td>Off</td><td>< 10 m</td></tr>
<tr><td>Off</td><td>On</td><td>< 10 us</td></tr>
<tr><td>On</td><td>On</td><td>< 10 us</td></tr>
<tr><td>On</td><td>Off</td><td>< 300 u</td></tr>
</table>

Exit latency to POWER_STATE_0 is measured from when the MAC changes the Power down</td>
<td>SATA</td>
</tr>
</table>

		value to when the PHY deasserts PHY status. The actual PHY latency must provide enough margin from the indicated limits to enable compliant device behavior per the SATA specification. A MAC must map the available PHY states to SATA states. Note: PLL shutdown is only possible if PowerDown is set to a state with PCLK off.	

<table>
<tr>
<td>PowerDown[2:0]
DisplayPort Mode</td>
<td>N/A</td>
<td>DisplayPort mode power states:

<table>
<tr><th>[3]</th><th>[2]</th><th>[1]</th><th>[0]</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>POWER_STATE_0 Operational state</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>POWER_STATE_1 Phy specific</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>POWER_STATE_2 Phy specific</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>POWER_STATE_3 Phy specific</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>POWER_STATE_4 Phy specific</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>POWER_STATE_5 Phy specific</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>POWER_STATE_6 Phy specific</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>POWER_STATE_7 Phy specific</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>POWER_STATE_8 Phy specific</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>POWER_STATE_9 Phy specific</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>POWER_STATE_10 Phy specific</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>POWER_STATE_11 Phy specific</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>POWER_STATE_12 Phy specific</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td><td>POWER_STATE_13 Phy specific</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td><td>POWER_STATE_14 Phy specific</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>POWER_STATE_15 Phy specific</td></tr>
</table>

A PIPE compliant DPRX PHY is recommended to support the following power states, although the mapping to the above power state encodings is PHY implementation specific:

<table>
<tr><td>Main Link RX</td><td>Aux Link</td><td>Exit Latency</td><td>Requ</td></tr>
<tr><td>Enabled</td><td>Enabled</td><td>N/A</td><td>Yes</td></tr>
<tr><td>Disabled</td><td>Enabled for differential signal monitoring</td><td><1ms</td><td>Yes</td></tr>
<tr><td>Disabled</td><td>Enabled for differential signal monitoring</td><td><80ms</td><td>No</td></tr>
<tr><td>Disabled</td><td>Enabled</td><td><0.5us</td><td>Yes</td></tr>
</table>
</td>
<td>DisplayPort</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td></td>
<td>
<table>
<tr><td></td><td></td><td></td><td>eDP only</td><td></td></tr>
<tr><td>Disabled</td><td>Enabled</td><td><20us</td><td>Yes for eDP only</td><td></td></tr>
</table>

A PIPE compliant DPTX PHY is recommended to support the following power states, although the mapping to specific power state encodings is PHY implementation specific: (TBD: may want to specify exit latency)
<table>
<tr><th>Main Link TX</th><th>Aux Link</th><th>DP_PWR</th></tr>
<tr><td>Enabled</td><td>Enabled</td><td>Enabled</td></tr>
<tr><td>Disabled</td><td>Enabled</td><td>Enabled</td></tr>
</table>
</td>
<td></td>
</tr>
<tr>
<td>RxEIDetectDisable</td>
<td>High</td>
<td>Optionally implemented by the PHY to facilitiate L1 substate management. When asserted, this signal asynchronously disables the receiver Electrical Idle detect logic, forcing the RxElecIdle PHY output to a value of ‘1’. If this signal transitions to deasserted after being asserted, the RxElecIdle output shall be forced to a high value until the Electrical Idle detect logic is functional. Note: The PHY may choose to support managing L1 substates via this signal and the TxCommonModeDisable signal instead of the PowerDown[3:0] signal.

This signal is only used by PHYs that support PCIe L1 substates.</td>
<td>PCIe</td>
</tr>
</table>

TxCommonModeDisable	High	Optionally implemented by the PHY to facilitate L1 substate management. When asserted, this signal asynchronously disables the transmitter DC common mode logic. Note: The PHY may choose to support managing L1 substates via this signal and the RxEIDetectDisable signal instead of the PowerDown[3:0] signal. This signal is only valid when PowerDown is at a value that supports L1 substate management via TxCommonModeDisable and RxEIDetectDisable. This signal is only used by PHYs that support PCIe L1 substates.	PCIe

<table>
<tr>
<td>Rate[1:0]</td>
<td>N/A</td>
<td>Control the link signaling rate.

PCI Express Mode:
<table>
<tr><td>Value</td><td>Description</td></tr>
<tr><td>0</td><td>Use 2.5 GT/s signaling rat</td></tr>
<tr><td>1</td><td>Use 5.0 GT/s signaling rat</td></tr>
<tr><td>2</td><td>Use 8.0 GT/s signaling rat</td></tr>
<tr><td>3</td><td>Use 16.0 GT/s signaling ra</td></tr>
</table>
Sata Mode:
<table>
<tr><td>Value</td><td>Description</td></tr>
<tr><td>0</td><td>Use 1.5 GT/s signaling rat</td></tr>
<tr><td>1</td><td>Use 3.0 GT/s signaling rat</td></tr>
<tr><td>2</td><td>Use 6.0 GT/s signaling rat</td></tr>
<tr><td>3</td><td>Reserved</td></tr>
</table>
USB Mode:
<table>
<tr><td>Value</td><td>Description</td></tr>
<tr><td>0</td><td>Use 5.0 GT/s signaling rat</td></tr>
<tr><td>1</td><td>Use 10.0 GT/s signaling ra</td></tr>
<tr><td>2</td><td>Reserved</td></tr>
<tr><td>3</td><td>Reserved</td></tr>
</table>
DisplayPort Mode:
<table>
<tr><td>Value</td><td>Description</td></tr>
<tr><td>0</td><td>Use 1.62 Gbps signaling rat</td></tr>
<tr><td>1</td><td>Use 2.7 Gbps signaling rate</td></tr>
<tr><td>2</td><td>Use 5.4 Gbps signaling rate</td></tr>
<tr><td>3</td><td>Use 8.1 Gbps signaling rate</td></tr>
</table>
Converged IO Mode:
<table>
<tr><td>Value</td><td>Description</td></tr>
<tr><td>0</td><td>Use 10.0 GT/s signaling ra</td></tr>
<tr><td>1</td><td>Use 20.0 GT/s signaling ra</td></tr>
<tr><td>2</td><td>Reserved</td></tr>
<tr><td>3</td><td>Reserved</td></tr>
</table>
PIPE implementations that only support one signaling rate do not implement this signal.</td>
<td>PCIe, SATA, USB, DisplayPort, Converged IO</td>
</tr>
</table>

<table>
<tr>
<td>Width[1:0]</td>
<td>N/A</td>
<td>Controls the PIPE data path width. For SerDes architecture, this applies only to the transmit side and RxWidth[1:0] controls the receive side.

If EncodeDecodeBypass is ‘0’
<table>
<tr><th>Value</th><th>Datapath Width</th></tr>
<tr><td>0</td><td>8 bits</td></tr>
<tr><td>1</td><td>16 bits</td></tr>
<tr><td>2</td><td>32 bits</td></tr>
<tr><td>3</td><td>Reserved</td></tr>
</table>
If EncodeDecodeBypass is ‘1’ or in SerDes architecture
<table>
<tr><th>Value</th><th>Datapath Width</th></tr>
<tr><td>0</td><td>10 bits</td></tr>
<tr><td>1</td><td>20 bits</td></tr>
<tr><td>2</td><td>40 bits</td></tr>
<tr><td>3</td><td>80 bits (PCIe SerDes only)</td></tr>
</table>
Note: PHYs that support greater than x4 link width must provide option of 32-bit data width or smaller.

PIPE implementations that only support one option at each signaling rate do not implement this signal.</td>
<td>PCIe, SATA, USB, DisplayPort</td>
</tr>
</table>

<table>
<tr>
<td>PCLK Rate[2:0]</td>
<td>N/A</td>
<td>Control the PIPE PCLK rate

SATA Mode:

0 37.5 Mhz
1 75 Mhz
2 150 Mhz
3 300 Mhz
4 600 Mhz
5 Reserved
6 Reserved
7 Reserved

PCI Express Mode:
0 62.5 Mhz
1 125 Mhz
2 250 Mhz
3 500 Mhz
4 1000 Mhz
5 2000 Mhz
6 Reserved
7 Reserved

USB Mode:
0 125 Mhz
1 250 Mhz
2 312.5 Mhz (10 GT/s)
3 500 Mhz
4 625 Mhz (10 GT/s)
5 1250 Mhz (10 GT/s)
6 Reserved
7 Reserved

DisplayPort Mode:
<table>
<tr><th>Value</th><th>Rate</th></tr>
<tr><td>0</td><td>40.5 Mhz</td></tr>
<tr><td>1</td><td>62.52 Mhz</td></tr>
<tr><td>2</td><td>81 Mhz</td></tr>
<tr><td>3</td><td>135 Mhz</td></tr>
<tr><td>4</td><td>162 Mhz</td></tr>
<tr><td>5</td><td>202.5 Mhz</td></tr>
<tr><td>6</td><td>270 Mhz</td></tr>
<tr><td>7</td><td>405 Mhz</td></tr>
<tr><td>8</td><td>540 Mhz</td></tr>
<tr><td>9</td><td>810 Mhz</td></tr>
<tr><td>10-15</td><td>Reserved</td></tr>
</table>
Converged IO Mode:
This signal is not used.</td>
<td>PCIe, SATA, USB, DisplayPort</td>
</tr>
</table>

<table>
<tr><td></td><td></td><td>PIPE implementations that do not support more than one PCLK rate for any analog signaling rate do not implement this signal.</td><td></td></tr>
<tr><td>RxPresetHint[2:0]</td><td>N/A</td><td>This signal has been deprecated and will be removed in the next rev of the spec.</td><td>PCIe</td></tr>
<tr><td>RXTermination</td><td>High</td><td>Controls presence of receiver terminations:
<table><tr><td>Value</td><td>Description</td></tr><tr><td>0</td><td>Terminations removed</td></tr><tr><td>1</td><td>Terminations present</td></tr></table>
Implementation of this signal is only required for PHYs that support USB mode; this signal is optional for other protocols.</td><td>USB, PCIe</td></tr>
</table>

RxStandby	N/A	SATA Mode: Controls whether the PHY RX is active when the PHY is in any power state with PCLK on. 0 – Active 1 – Standby RxStandby is ignored when the PHY is in any power state where the high speed receiver is always off. PCI Express Mode: Controls whether the PHY RX is active when the PHY is in P0 or P0s. 0 – Active 1 – Standby RxStandby is ignored when the PHY is in P1 or P2. USB Mode and Converged IO Mode: Controls whether the PHY RX is active when the PHY is in any power state with PCLK on. 0 – Active 1 – Standby RxStandby is ignored when the PHY is in any power state where the high speed receiver is always off.	PCIe, SATA, USB, Converged IO

Table 6-6. Command Interface Output Signals

Name	Active Level	Description	Relevant Protocols
RefClkRequired#	N/A	This signal is deasserted by the PHY when the reference clock can be safely removed in low power states. This signal shall remain asserted low in all states except P2 and PowerDown states assigned to L1 substate support. While in P2 or L1 substate PowerDown states, the PHY deasserts this signal when it is ready for reference clock removal. While in P2 or L1 substate PowerDown states, the PHY asserts this signal when it detects a P2 or L1 substate exit request.	PCIe
RxStandbyStatus	N/A	SATA Mode and PCI Express Mode and Converged IO Mode: The PHY uses this signal to indicate its RxStandby state. 0 – Active 1 – Standby RxStandbyStatus reflects the state of the high speed receiver. The high speed receiver is always off in PHY states that do not provide PCLK. PCI Express Mode: RxStandbyStatus is undefined when the power state is P1 or P2. This signal is not applicable to USB mode.	PCIe, SATA, Converged IO

6.1.3 Status Interface

Table 6-7. Status Interface Input Signals

Name	Active Level	Description	Relevant Protocols
PclkChangeAck	High	Only used when PCLK is a PHY input. Asserted by the MAC when a PCLK rate change is complete and stable. After the MAC asserts PclkChangeAck the PHY responds by asserting PhyStatus for one cyle and de-asserts PclkChangeOk at the same time as PhyStatus. The controller shall deassert PclkChangeAck when PclkChangeOk is sampled low. This signal is not used by any PhyMode that does not perform dynamic rate changes.	PCIe, SATA, USB
AsyncPowerChangeAck	High	Only used when transitioning between two power states without PCLK. After the PHY asserts PhyStatus to acknowledge the power state change the MAC responds by asserting AsyncPowerChangeAck until it samples PhyStatus deasserted. Implementation of this signal is only required for PHYs that support PCI Express L1 PM Substates managed via the PowerDown signal.	PCIe

Table 6-8. Status Interface Output Signals

Name	Active Level	Description	Relevant Protocols

RxValid	High	Indicates symbol lock and valid data on *RxData* and *RxDataK* and further qualifies RxDataValid when used. PCI Express Mode at 8 GT/s and 16 GT/s and USB Mode at 10 GT/s only: When BlockAlignControl=1: - RxValid indicates that the block aligner is conceptually in the "Aligned" state (see PCI Express or USB 3.1 Spec) - If the block aligner transitions "Aligned" -> "Unaligned" state RxValid can deassert anywhere within a block - If the block aligner transitions "Unaligned" -> "Aligned" state RxValid is asserted at the start of a block Note that a PHY is not required to force its block aligner to the unaligned state when BlockAlignControl transitions to one. When BlockAlignControl=0: - RxValid is constantly high indicating that the block aligner is conceptually in the "Locked" state (see PCI Express or USB 3.1 Spec). RxValid can be dropped on detecting and elastic buffer underflow or overflow. If de-asserted it must not re-assert while *BlockAlignControl* is de-asserted. In the SerDes architecture, RxValid is used to indicate that the recovered clock is stable. The MAC can start symbol or block lock after RxValid is asserted.	PCIe, USB, SATA DisplayPort RX

PhyStatus	High	Used to communicate completion of several PHY functions including stable PCLK and/or Max PCLK (depending on clocking mode) after Reset# deassertion, power management state transitions, rate change, and receiver detection. When this signal transitions during entry and exit from any PHY state where PCLK is not provided, then the signaling is asynchronous. In error situations (where the PHY fails to assert PhyStatus) the MAC can take MAC-specific error recovery actions.	PCIe, SATA, USB, DisplayPort, Converged IO
RxElecIdle	High	Indicates receiver detection of an electrical idle. While deasserted with the PHY in P2 (PCI Express mode) or the PHY in P0, P1, P2, or P3 (USB Mode and Converged IO Mode), indicates detection of either: PCI Express Mode: a beacon. USB Mode and Converged IO Mode : LFPS This is an asynchronous signal. See RxEIDetectDisable for additional information. PCI Express Mode: It is required at the 5.0 GT/s, 8.0 GT/s and 16 GT/s rates that a MAC uses logic to detect electrical idle entry instead of relying on the RxElecIdle signal. Sata Mode: The time the signal is asserted must match the actual idle time on the analog bus within -16/+0 ns.	PCIe, SATA, USB, Converged IO

<table>
<tr>
<td>RxStatus[2:0]</td>
<td>N/A</td>
<td>Encodes receiver status and error codes for the received data stream when receiving data.
<table>
<tr><th>[2]</th><th>[1]</th><th>[0]</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Received data OK</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>PCI Express Mode: 1 SKP added
USB Mode: 1 SKP Ordered Set added
Sata Mode: 1 ALIGN added
Asserted with first byte of Align that was added. An align may only be added in conjunction with receiving one or more aligns in the data stream and only when the elasticity buffer is operating in half full mode</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>PCI Express Mode: 1 SKP removed
USB Mode: 1 SKP Ordered Set removed
SATA Mode: 1 or more ALIGNs removed
This status is asserted with first non ALIGN byte following an ALIGN. This status message is applicable to both EB buffer modes.</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>PCI Express and USB Modes:
Receiver detected
SATA Mode: Misalign
Signaled on the first symbol of an ALIGN that was received misaligned in elasticity buffer nominal half full mode. Signaled on the first data following an align in elasticity buffer nominal empty mode.</td></tr>
</table>
</td>
<td>PCIe, SATA, USB</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td></td>
<td>
<table>
<tr><td>1</td><td>0</td><td>0</td><td>Both 8B/10B (128B/130B[1]) decode error and (optionally) Receive Disparity error
Note: This error is never reported if EncodeDecodeBypass is asserted.</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Elastic Buffer overflow</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Elastic Buffer underflow.
This error code is not used if the elasticity buffer is operating in the nominal buffer empty mode.</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Receive disparity error (Reserved if Receive Disparity error is reported with code 0b100)
Not used if EncodeDecodeBypass is asserted.
For USB3 Gen2, indicates “SKP Corrected”.</td></tr>
</table>

The only status applicable to SerDes architecture is ‘Receiver detected’ (0x3).
</td>
<td></td>
</tr>
<tr>
<td>PowerPresent</td>
<td>High</td>
<td>USB Mode: Indicates the presence of VBUS.
Implementation of this signal is only required for PHYs that support USB mode.</td>
<td>USB</td>
</tr>
<tr>
<td>PclkChangeOk</td>
<td>High</td>
<td>Only used when PCLK is a PHY input. Asserted by the PHY when it is ready for the MAC to change the PCLK rate. The PHY shall only assert this signal after the MAC has requested a PCLK rate change by changing PCLK_Rate.

This signal is not used for DisplayPort or Converged IO Mode.</td>
<td>PCIe, SATA, USB</td>
</tr>
</table>

[1] Disparity errors are not reported when the rate is 8.0 GT/s or 16 GT/s.

6.1.4 Message Bus Interface

The message bus interface provides a way to initiate and participate in non-latency sensitive PIPE operations using a small number of wires, and it enables future PIPE operations to be added without adding additional wires. The use of this interface requires the device to be in a power state with PCLK running. Control and status bits used for PIPE operations are mapped into 8-bit registers that are hosted in 12-bit address spaces in the PHY and the MAC. The registers are accessed via read and write commands driven over the signals listed in Table 6-9. These signals are synchronous with PCLK and are reset with Reset#. The specific commands and framing of the transactions sent over the message bus interface are described in the following subsections.

Table 6-9 Message Bus Interface Signals

Name	Direction	Description
M2P_MessageBus[7:0]	Input	The MAC multiplexes command, any required address, and any required data for sending read and write requests to access PHY PIPE registers and for sending read completion responses and write ack responses to PHY initiated requests.
P2M_MessageBus[7:0]	Output	The PHY multiplexes command, any required address, and any required data for sending read and write requests to access MAC PIPE registers and for sending read completion responses and write ack responses to MAC initiated requests.

6.1.4.1 Message Bus Interface Commands

The 4-bit commands used for accessing the PIPE registers across the message bus are defined in Table 6-10. A transaction consists of a command and any associated address and data, as specified in the table. The table also specifies the number of PCLK cycles that it takes to transfer the transaction across the message bus interface. The order in which the bits are transferred across the interface are illustrated in Figure 6-1, Figure 6-2, Figure 6-3, and Figure 6-4.

To address the case where multiple PIPE interface signals can change on the same PCLK, the concept of write_uncommitted and write_committed is introduced. A series of write_uncommitted transactions followed by one write_committed transaction provides a mechanism by which all the uncommitted writes and the final committed write are executed in an atomic manner, thus taking effect during the same PCLK cycle.

Errors in SKP ordered sets shall be reported by the PHY as 128/130 decode errors. An error in a SKP ordered set shall be reported if there is an error in the first 4N+1 symbols of the skip ordered set.

To enable the write_uncommitted command, designs must implement a write buffer in the PHY and the MAC, where each write buffer entry can accommodate the three bytes worth of information associated with each write transaction. The minimum write buffer depth required is five; however, this number may increase in the future when new PIPE operations are mapped into the message bus interface.

Table 6-10 Message Bus Commands

Encoding	Command	Description	Required Fields	Cycles to Transmit
4'b0000	NOP	Idle. See Figure 6-1.	Command[3:0]	1
4'b0001	write_uncommitted	The current write should be saved off into a write buffer and its associated data values are updated into the relevant PIPE register at a future time when a write_committed is received. This is useful for signals that must change in the same cycle but that are distributed among multiple registers. See Figure 6-4.	Command[3:0], Address[11:0], Data[7:0]	3
4'b0010	write_committed	The current write as well as any previously uncommitted writes saved into the write buffer should be committed, i.e. their values should be updated into the PIPE registers. Once a write_committed is sent, no new writes, whether committed or uncommitted, may be sent until a write_ack is received. See Figure 6-4.	Command[3:0], Address[11:0], Data[7:0]	3
4'b0011	read	Used to read contents of a PIPE register. Only one read can be outstanding at a time in each direction. See Figure 6-2.	Command[3:0], Address[11:0]	2
4'b0100	read completion	Data response to a read. See Figure 6-3.	Command[3:0], Data[7:0]	2
4'b0101	write_ack	Used to acknowledge receipt of a write_committed and readiness to accept another write. The ack is sent when the write buffer is flushed and the resulting PIPE	Command[3:0]	1

		operation is guaranteed to start in a deterministic amount of time. Note: This does not provide confirmation that the PIPE operation triggered by the write has completed. See Figure 6-1.		
All others	Reserved	N/A	N/A	N/A

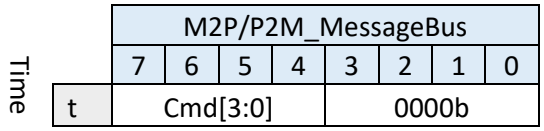


Figure 6-1. Command Only Message Bus Transaction Timing (NOP, write_ack)

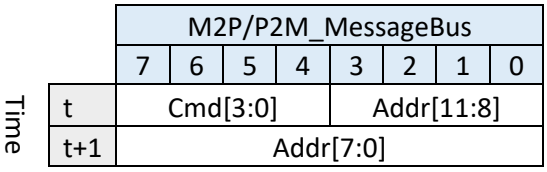


Figure 6-2. Command+Address Message Bus Transaction Timing (Read)

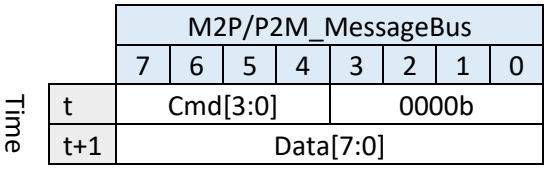
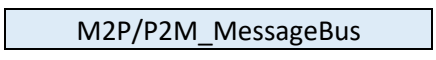


Figure 6-3. Command+Data Message Bus Transaction Timing (Read completion)



	7	6	5	4	3	2	1	0
t	Cmd[3:0]				Addr[11:8]			
t+1	Addr[7:0]							
t+2	Data[7:0]							

Time

Figure 6-4. Command+Address+Data Message Bus Transaction Timing (Write_uncommitted, Write_committed)

6.1.4.2 Message Bus Interface Framing

The framing of transactions is implicitly derived by adhering to the following rules:

1. All zeroes must be driven on the message bus when idle.
2. An idle to non-idle transition indicates the start of a transaction; a new transaction can start immediately the cycle after the end of the previous transaction without an intervening idle.
3. The number of cycles to transmit a transaction depends on the command and is specified in Table 6-10.
4. The cycles associated with one transaction must be transferred in contiguous cycles.

Figure 6-5 illustrates the framing of a couple of transactions on the message bus. The start of the first transaction is inferred by the idle to non-idle transition. The command is decoded as a write, which takes three cycles to transmit. Since the cycle following the end of the write is non-idle, it is inferred to be the start of the next transaction, which is decoded to be another write that takes three cycles to transmit.

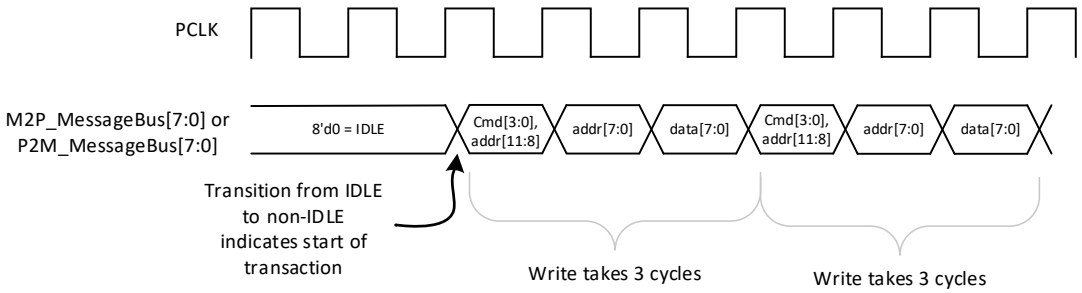


Figure 6-5. Message Bus Transaction Framing

6.2 PHY/MAC Interface Signals – SerDes Architecture Only

This section describes any signals for SerDes architecture that are required in addition to those defined in section 6.1.

6.2.1 Data Interface

Table 6-11. SerDes Only: Receive Data Interface Output Signals

Name	Active Level	Description	Relevant Protocols
RxCLK	Rising Edge	*This clock signal is only used in the SerDes architecture.* Recovered clock used for RxData in the SerDes architecture.	PCIe, USB, DisplayPort RX, Converged IO

6.2.2 Command Interface

Table 6-12. SerDes Only: Command Interface Input Signals

Name	Active Level	Description	Relevant Protocols
RxWidth[1:0]	N/A	*This clock signal is only used in the SerDes architecture.* Controls the PIPE receive data path width If EncodeDecodeBypass is '0' Value | Datapath Width 0 | 8 bits 1 | 16 bits 2 | 32 bits 3 | Reserved If EncodeDecodeBypass is '1' or in SerDes architecture Value | Datapath Width 0 | 10 bits 1 | 20 bits 2 | 40 bits 3 | 80 bits (PCIe SerDes only) Note: PHYs that support greater than x4 link width must provide option of 32-bit data width or smaller. PIPE implementations that only support one option at each signaling rate do not implement this signal.	PCIe, SATA, USB, DisplayPort

6.3 PHY/MAC Interface Signals – Original PIPE Only

This section describes signals for Original PIPE that are required in addition to those define in section 6.1.

6.3.1 Data Interface

Table 6-13. Original PIPE Only: Transmit Data Interface Input Signals

Name	Active Level	Description	Relevant Protocols
TxDataK[7:0] for 64-bit interface TxDataK[3:0] for 32-bit interface TxDataK[1:0]for 16-bit interface TxDataK for 8-bit interface	N/A	*This signal is not used in the SerDes architecture.* Data/Control for the symbols of transmit data. For 64-bit interfaces, Bit 0 corresponds to the low-byte of TxData and bit 7 corresponds to the upper byte. For 32-bit interfaces, Bit 0 corresponds to the low-byte of TxData, Bit3 corresponds to the upper byte. For 16-bit interfaces, Bit 0 corresponds to the low-byte of TxData, Bit 1 to the upper byte. A value of zero indicates a data byte, a value of 1 indicates a control byte. Not used in PCI Express mode at 8 GT/s or 16 GT/s. Not used in USB mode at 10 GT/s. Not used in Converged IO mode.	PCIe, SATA, USB
TxStartBlock	N/A	*This signal is not used in the SerDes architecture.* PCI Express Mode and USB Mode: Only used at the 8.0 GT/s and 16 GT/s PCI Express signaling rates and the 10 GT/s USB signaling rate. This signals allow the MAC to tell the PHY the starting byte for a 128b block. The starting byte for a 128b block must always start with byte 0 of the data interface.	PCIe, USB

Table 6-14. Original PIPE Only: Receive Data Interface Output Signals

Name	Active Level	Description	Relevant Protocols
RxDataK[7:0] for 64-bit interface RxDataK[3:0] for 32-bit interface RxDataK[1:0] for 16-bit interface RxDataK for 8-bit interface	N/A	*This signal is not used in the SerDes architecture.* Data/Control bit for the symbols of receive data. For 64-bit interfaces, Bit 0 corresponds to the low-byte of TxData and bit 7 corresponds to the upper byte. For 32-bit interfaces, Bit 0 corresponds to the low-byte of RxData, Bit3 corresponds to the upper byte. For 16-bit interface, Bit 0 corresponds to the low-byte of RxData[15:0], Bit 1 to the upper byte. A value of zero indicates a data byte; a value of 1 indicates a control byte. Not used in PCI Express mode at 8 GT/s or 16 GT/s or USB mode at 10 GT/s or Converged IO mode. When the PHY is in a SATA mode, the first valid data following an ALIGN primitive must appear as byte 0 in the receive data.	PCIe, SATA, USB
RxDataValid	N/A	*This signal is not used in the SerDes architecture.* PCI Express Mode and SATA Mode and USB Mode: This signal allows the PHY to instruct the MAC to ignore the data interface for one clock cycle. A value of one indicates the MAC will use the data, a value of zero indicates the MAC will not use the data. RxDataValid shall not assert when RXvalid is de-asserted in PHY modes that require the use of RxDataValid. If a PHY supports the RxDataValid signal it shall keep RxDataValid asserted when the PHY is in a mode that does not require the signal. The MAC may ignore RxDataValid when it is in a mode that does not require the signal.	PCIe, SATA, USB

RxStartBlock	N/A	*This signal is not used in the SerDes architecture.* PCI Express Mode and USB Mode: Only used at the 8.0 GT/s and 16 GT/s PCI Express signaling rates and the 10 GT/s USB signaling rate. This signal allows the PHY to tell the MAC the starting byte for a 128b block. The starting byte for a 128b block must always start with byte 0 of the data interface. Note: If there is an invalid sync header decoded on RxSyncHeader[3:0] and block alignment is still present (RxValid == 1), then the PHY will assert RxStartBlock with the invalid sync header on RxSyncHeader[3:0] RxStartBlock shall not assert when RxValid is de-asserted	PCIe, USB

6.3.2 Command Interface

Table 6-15. Command Interface Input Signals

Name	Active Level	Description	Relevant Protocols
TxCompliance	High	*This signal is not used in the SerDes architecture.* PCI Express Mode: Sets the running disparity to negative. Used when transmitting the PCI Express compliance pattern. Implementation of this signal is only required for PHYs that support PCI Express mode. This signal is sampled by TxDataValid.	PCIe
TxSyncHeader[3:0]	N/A	*This signal is not used in the SerDes architecture.* PCI Express Mode: Only the lower two bits ([1:0]) are utilized. Provides the sync header for the PHY to use in the next 130b block. The PHY reads this value when the TXStartBlock signal is asserted. This signal is only used at the 8.0 GT/s and 16 GT/s signaling rates. USB Mode: Provides the sync header for the PHY to use in the next 132b block. The PHY reads this value when the TXStartBlock signal is asserted. This signal is only used at the 10 GT/s signaling rate.	PCIe, USB

Table 6-16. Original PIPE Only: Command Interface Output Signals

Name	Active Level	Description	Relevant Protocols
RxSyncHeader[3:0]	N/A	*This signal is not used in the SerDes architecture.* PCI Express Mode: Only the lower two bits ([1:0]) are utilized. Provides the sync header for the MAC to use with the next 128b block. The MAC reads this value when the RxStartBlock signal is asserted. This signal is only used at the 8.0 GT/s and 16 GT/s signaling rates. USB Mode: Provides the sync header for the MAC to use with the next 128b block. The MAC reads this value when the RxStartBlock signal is asserted. This signal is only used at the 10.0 GT/s signaling rate. Note: The PHY shall pass blocks and headers normally across the PIPE interface even if the decoded SyncHeader is invalid.	PCIe, USB

Table 6-17. Original PIPE only: Status Interface Output Signals

Name	Active Level	Description	Relevant Protocols
AlignDetect	High	*This signal is not used in the SerDes architecture.* Indicates receiver detection of an Align. A PHY is only required to assert this signal when the Elasticity Buffer is running in nominal empty mode. The PHY shall only toggle this signal after obtaining bit and symbol lock. Each ALIGN received shall map to AlignDetect being asserted for one PCLK. The spacing between PCLK pulses for ALIGNs should map analog spacing of received ALIGNs as closely as possible. However there is no guarantee to have PCLK domain spacing between back to back AlignDetect pulses match the analog spacing exactly due to differences in the receive clock domain and the PCLK domain. For example: 1.5 GT/s with 8-bit data path PCLK=150MHz, the nominal spacing is 4 PCLK's. 3.0 GT/s with 8-bit data path PCLK=300MHz, the nominal spacing is 4 PCLK's. 6.0 GT/s with 16-bit data path PCLK=300MHz, the nominal spacing is every other PCLK. Due to differences in the PCLK and receive clocks, the nominal spacing can be off by one PCLK in either direction. In the example with PCLK rate being equal to Gen3 received clock rate, clock domain crossing could lead to AlignDetect being asserted for consecutive PCLK cycles without gap.	SATA

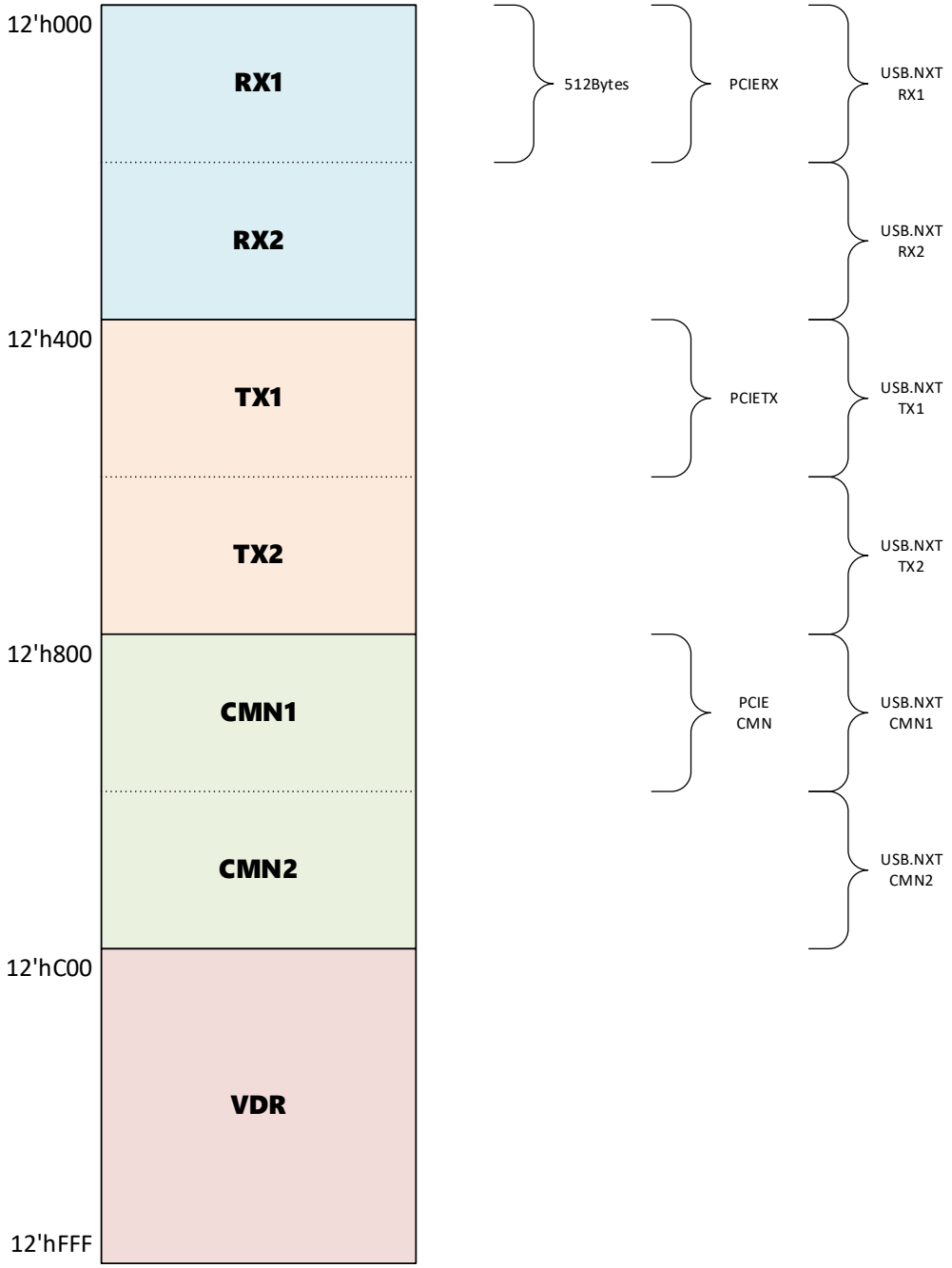
7 PIPE Message Bus Address Spaces

The PIPE specification defines 12-bit address spaces to enable the message bus interface; the MAC and the PHY each implement unique 12-bit address spaces as shown in Figure 7-1. These address spaces are used to host registers associated with certain PIPE operations. The MAC and PHY access specific bits in the registers to initiate operations, to participate in handshakes, or to indicate status. The MAC initiates requests on the message bus interface to access registers hosted in the PHY address space. The PHY initiates requests on the message bus interface to access registers hosted in the MAC address space.

Each 12-bit address space is divided into four main regions: receiver address region, transmitter address region, common address region, and vendor specific address region. The receiver address region is used to configure and report status related to receiver operation; it spans the 1024KB region from 12'h000 to 12'h3FF and supports up to two receivers with 512KB allocated to each. The transmitter address region is used to configure and report status related to transmitter operation; it spans the 1024KB region from 12'h400 to 12'h7FF and supports up to two transmitters, TX1 and TX2, with a 512KB region associated with each. The common address region hosts registers relevant to both receiver and transmitter operation; it spans the 1024KB region from 12'h800 to 12'hBFF and supports up two sets of Rx/Tx pairs with 512KB allocated toward the common registers for each pair. The vendor specific address region is the 1024K region from 12'hC00 to 12'hFFF and enables individual vendors to define registers as needed outside of those defined in this PIPE specification.

As noted above, the address space is defined to support configurable Rx/Tx pairs. Up to two differential pairs are assumed to be operational at any one time. Supported combinations are one Rx and one Tx pair, two Tx pairs, or two Rx pairs.

Figure 7-1. Message Bus Address Space



The PCIe RX margining operations and elastic buffer depth are controlled via registers hosted in these address spaces. Additionally, several legacy pipe control and status signals have been mapped into registers hosted in these address spaces.

The following subsections define the PHY registers and the MAC registers. Individual register fields are specified as required or optional. In addition, each field has an attribute description of either level or 1-cycle assertion. When a level field is written, the value written is maintained by the hardware until the next write to that field or until a reset occurs. When a 1-cycle field is written to assert the value high, the hardware maintains the assertion for only a single cycle and then automatically resets the value to zero on the next cycle.

7.1 PHY Registers

Table 7-1 lists the PHY registers and their associated address. The details of each register are provided in the subsections below.

To support configurable pairs, the same registers defined for RX1 are also defined for RX2, the same registers defined for TX1 are defined for TX2, and the same registers defined for CMN1 are defined for CMN2. Only two differential pairs are active at a time based on configuration; valid combinations correspond to registers defined in RX1+TX1+CMN1, RX1+RX2+CMN1+CMN2, or TX1+TX2+CMN1+CMN2.

A PHY that does not support configurable pairs only implements registers defined for RX1, TX1, and CMN1.

Table 7-1 PHY RX Registers

Byte Address	Register Name	Notes
12'h0	RX1: RX Margin Control0	
12'h1	RX1: RX Margin Control1	
12'h2	RX1: Elastic Buffer Control	Original PIPE only
12'h3	RX1: PHY RX Control0	Original PIPE only
12'h4	RX1: PHY RX Control1	
12'h5	RX1: PHY RX Control2	
12'h6	RX1: PHY RX Control3	
12'h7-12'h1FF	RX1: Reserved	
12'h200 to 12'h3FF	RX2: Same registers are defined in this region for RX2 as for RX1 above.	
12'h400	TX1: PHY TX Control0	Original PIPE only
12'h401	TX1: PHY TX Control1	Original PIPE only
12'h402	TX1: PHY TX Control2	
12'h403	TX1: PHY TX Control3	
12'h404	TX1: PHY TX Control4	
12'h405	TX1: PHY TX Control5	
12'h406	TX1: PHY TX Control6	
12'h407	TX1: PHY TX Control7	
12'h408	TX1: PHY TX Control8	
12'h409	TX1: PHY TX Control9	Original PIPE only
12'h410-12'h5FF	TX1: Reserved	
12'h600-12'h7FF	TX2: Same registers are defined in this region for TX2 as for TX1 above	
12'h800	CMN1: PHY Common Control0	Original PIPE only
12'h801-12'h9FF	CMN1: Reserved	
12'hA00 – 12'BFF	CMN2: Same registers are defined in this region for CMN2 as for CMN1 above	
12'hC00-12'hFFF	VDR: Reserved	

7.1.1 Address 0h: RX Margin Control0

This register is used along with RX Margin Control1 to control PCIe Lane Margining at the Receiver.

Bit	Default	Attribute	Required	Description
[7:4]	0h	N/A	N/A	Reserved
[3]	0h	1-cycle	PCIe (Optional)	**Sample Count Reset** – This field is used to reset the 'Sample Count[6:0]' field of the RX Margin Status1 register.
[2]	0h	1-cycle	PCIe (Optional)	**Error Count Reset** – This field is used to reset the 'Error Count[5:0]' field of the RX Margin Status2 register.
[1]	0h	Level	PCIe	**Margin Voltage or Timing** – This field is used to select between margining voltage (1'b0) or margining timing (1'b1). The value can be changed only when margining is stopped.
[0]	0h	Level	PCIe	**Start Margin** – This field is used to start and stop margining. A transition from 1'b0 to 1'b1 starts the margining process. A transition from 1'b1 to 1'b0 stops the margining process.

7.1.2 Address 1h: RX Margin Control1

This register is used along with RX Margin Control0 to control PCIe RX margining.

Bit	Default	Attribute	Required	Description
[7]	0h	Level	PCIe	**Margin Direction** – This field is used to control time or voltage direction for margining. For timing margining, this field steps time left (1'b0) or right (1'b1). For voltage margining, this field steps voltage up (1'b0) or down (1'b1). This value can be changed only when margining is stopped.
[6:0]	0h	Level	PCIe	**Margin Offset** – This field is used to change the margin offset a number of steps from the default position. This value can be changed even during the margining process.

7.1.3 Address 2h: Elastic Buffer Control

This register is used to control the elastic buffer depth, enabling the controller to optimize latency in nominal half full mode. The ability to control elastic buffer depth is an optional feature that may be especially beneficial for retimers operating in PCIe SRIS mode.

Bit	Default	Attribute	Required	Description
[7:0]	0h	Level	PCIe (optional)	**Elastic Buffer Depth Control** – This field is used to set the elastic buffer depth. The MAC must choose from the supported values advertised in the PHY datasheet. This value can only be changed during transmission of TS1 ordered sets. The PHY performs the adjustment as quickly as possible without waiting for SKPs. The PHY signals

				completion of elastic buffer depth adjustment by setting the Elastic Buffer Status register. Note: This field is not used in the SerDes architecture.

7.1.4 Address 3h: PHY RX Control0

This register is used to control receiver functionality.

Bit	Default	Attribute	Required	Description
[7:2]	0h	N/A	N/A	Reserved
[1]	0h	Level	PCIe, USB, SATA, Converged IO	**RxPolarity** -- This field is used to control polarity inversion on the received data. Value: 0 – PHY does no polarity inversion; 1 – PHY does polarity inversion Note: This field is not used in the SerDes architecture.
[0]	0h	Level	PCIe (optional), SATA (optional), USB (optional)	**Elasticity Buffer Mode** -- This field is used to select the Elasticity Buffer operating mode. Value: 0 – Nominal Half Full Buffer mode; 1 – Nominal Empty Buffer Mode This field can only be changed when the receiver is OFF and Pclk is running, e.g. P0 with RXStandby asserted or P1. This field is not valid when TxDetectRx/Loopback is asserted. The PHY is responsible for switching to Nominal Half Full Buffer mode when loopback slave is requested. The PCS is responsible for making stream switch and abiding by PCIE base spec rules for slave loopback stream switching, e.g. switch on 10b boundary in 8b/10b modes, etc. Note: This field is not used in the SerDes architecture.

7.1.5 Address 4h: PHY RX Control1

This register is used to control receiver functionality.

Bit	Default	Attribute	Required	Description
[7:1]	0h	N/A	N/A	Reserved
[0]	0h	Level	USB	**RxEqTraining** – This field is set to 1’b1 to instruct the receiver to bypass normal operation to perform equalization training. While performing training the

				state of the RxData interface is undefined.

7.1.6 Address 5h: PHY RX Control2

This register is used to control receiver functionality.

Bit	Default	Attribute	Required	Description
[7:3]	0h	N/A	N/A	Reserved
[2:0]	0h	N/A	N/A	Reserved

7.1.7 Address 6h: PHY RX Control3

This register is used to control receiver functionality.

Bit	Default	Attribute	Required	Description
[7:3]	0h	N/A	N/A	Reserved
[2]	0h	Level	PCIe	**InvalidRequest –** This field is used to indicate that the Link Evaluation feedback requested a link partner TX EQ setting that was out of range. The MAC sets this bit to '1' when it detects an out of range error locally based on calculated link partner transmitter coefficients based on the last valid link equalization feedback or it receives a NACK response from the link partner. The MAC resets this bit to '0' the next time it asserts RxEQEval. When a MAC sets this bit, it shall subsequently ask the PHY to perform an RxEQ evaluation using the last valid setting a second time. This field is only applicable at the 8.0 GT/s and 16 GT/s signaling rates.
[1]	0h	Level	PCIe, Converged IO, DisplayPort RX (optional)	**RxEqInProgress –** This field is used by the MAC to indicate when link equalization evaluation is in progress. The PHY may optionally use this field to enable and disable functionality that is only needed during link equalization evaluations. For PCIe: The MAC sets this bit to '1' at the same time as it sets RxEqEval to '1' in phase 2 or phase 3 of the link equalization process. The MAC resets this bit to '0' at the end of phase 2 or phase 3.
[0]	0h	Level	PCIe, Converged IO, DisplayPort RX (optional)	**RxEqEval --** This field is set to '1' by the MAC to instruct the PHY to start evaluation of the far end transmitter TX EQ settings.

				For PCI Express, this field is only used at the 8.0 GT/s and 16 GT/s signaling rates.

7.1.8 Address 400h: PHY TX Control0

This register is used to control transmitter functionality.

Bit	Default	Attribute	Required	Description
[7:2]	0h	N/A	N/A	Reserved
[1:0]	0h	Level	SATA	**TX Pattern[1:0] –** This field controls which pattern the PHY sends at the Gen 1 rate when sending OOB or initialization signaling. The PHY transmits this pattern at the Gen 1 rate regardless of what rate the PHY is configured at. 0 ALIGN 1 D24.3 2 D10.2 3 Reserved See Section 8.24 for a more detailed description of the usage of these pins. Note: This field is not used in the SerDes architecture.

7.1.9 Address 401h: PHY TX Control1

This register is used to control transmitter functionality.

Bit	Default	Attribute	Required	Description
[7:1]	0h	N/A	N/A	Reserved
[0]	0h	Level	USB	**TxOnesZeros –** This field is used when transmitting USB compliance patterns CP7 or CP8. When this field is set, the transmitter to transmit an alternating sequence of 50-250 ones and 50-250 zeros – regardless of the state of the TxData interface. This field is only applicable to 8b/10b modes. Note: This field is not used in the SerDes architecture.

7.1.10 Address 402h: PHY TX Control2

This register is used to control transmitter functionality.

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	1h	Level	PCIe, USB,	**TxDeemph[5:0**] – This field is part of TxDeemph[17:0], which selects transmitter de-

<table>
<tr>
<td></td><td></td><td></td><td>Converged IO</td>
<td>emphasis.

PCI Express Mode, when the rate is 2.5 or 5.0 GT/s:

<table>
<tr><td>Value</td><td>Description</td></tr>
<tr><td>0</td><td>-6dB de-emphasis</td></tr>
<tr><td>1</td><td>-3.5dB de-emphasis</td></tr>
<tr><td>2</td><td>No de-emphasis</td></tr>
<tr><td>3</td><td>Reserved</td></tr>
</table>

PIPE implementations that only support 2.5 GT/s do not implement this field. PIPE PHY implementations that do not support low swing are not required to support the no-de-emphasis mode.

PCI Express Mode, when the rate is 8.0 GT/s or 16 GT/s:
[5:0] C-1
[11:6] C0
[17:12] C+1

USB Mode, when the rate is 10.0 GT/s:
[5:0] C-1
[11:6] C0
[17:12] C+1

The field is not defined for USB Mode when the rate is 5.0 GT/s

Converged IO Mode, when the rate is 10 GT/s or 20 GT/s:
[5:0] C-1
[11:6] C0
[17:12] C+1

Note: The MAC must ensure that only supported values are used for TxDeemph. In cases where the implementation is required to keep track of TX coefficients from previous states, this shall be done by the MAC.</td>
</tr>
</table>

7.1.11 Address 403h: PHY TX Control3

This register is used to control transmitter functionality.

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	Level	PCIe, USB, Converged IO	**TxDeemph[11:6] --** This field is part of TxDeemph[17:0], which selects transmitter de-

				emphasis. See TxDeemph[5:0] for detailed description.

7.1.12 Address 404h: PHY TX Control4

This register is used to control transmitter functionality.

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	Level	PCIe, USB, Converged IO	**TxDeemph[17:12] --** This field is part of TxDeemph[17:0], which selects transmitter de-emphasis. See TxDeemph[5:0] for detailed description.

7.1.13 Address 405h: PHY TX Control5

This register is used to control transmitter functionality.

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5]	0h	1-cycle	PCIe	**GetLocalPresetCoefficients –** This field is used to request a preset to co-efficient mapping for the preset on LocalPresetIndex[3:0] to coefficients on LocalTxPresetCoefficient[17:0] Maximum Response time of PHY is 128 nSec. Note. A MAC can make this request any time after reset. Note. After a local preset coefficient request a MAC could set GetLocalPresetCoefficients again as soon as the next PCLK after the coefficients are driven on LocalTxPresetCoefficient[17:0]. This field is only used with a PHY that requires dynamic preset coefficient updates
[4:0]	0h	Level	PCIe	**LocalPresetIndex[4:0] –** This field is used to indicate the index for the local PHY preset coefficients requested by the MAC. The preset index value is encoded as follows: 00000b – 8 GT/s Preset P0. 00001b – 8 GT/s Preset P1. 00010b – 8 GT/s Preset P2. 00011b – 8 GT/s Preset P3. 00100b – 8 GT/s Preset P4. 00101b – 8 GT/s Preset P5. 00110b – 8 GT/s Preset P6.

				00111b – 8 GT/s Preset P7. 01000b – 8 GT/s Preset P8. 01001b – 8 GT/s Preset P9. 01010b – 8 GT/s Preset P10. 01011b – 16 GT/s Preset P0 01100b – 16 GT/s Preset P1 01101b – 16 GT/s Preset P2 01110b – 16 GT/s Preset P3 01111b – 16 GT/s Preset P4 10000b – 16 GT/s Preset P5 10001b – 16 GT/s Preset P6 10010b – 16 GT/s Preset P7 10011b – 16 GT/s Preset P8 10100b – 16 GT/s Preset P9 10101b – 16 GT/s Preset P10 All others -- Reserved This field is only used with a PHY that requires dynamic preset coefficient updates.

7.1.14 Address 406h: PHY TX Control6

This register is used to control transmitter functionality.

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	Level	PCIe	**FS[5:0] --** This field reflects the the FS value advertised by the link partner. The MAC shall only change this value when a new FS value is captured during link training. A PHY may optionally consider this value when deciding how long to evaluate TX equalization settings of the link partner. The MAC shall only change this field when a new FS value is captured during link training or if there is a rate change. The MAC shall drive the relevant 8 GT/s values when the operational rate is 8 GT/s, and it shall drive the relevant 16 GT/s values when the operational rate is 16 GT/s.

7.1.15 Address 407h: PHY TX Control7

This register is used to control transmitter functionality.

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	Level	PCIe	**LF[5:0] –** This field reflects the LF value advertised by the link partner. The MAC shall only change this value when a new LF value is captured during link training or when there is a rate change. A PHY may

				optionally consider this value when deciding how long to evaluate TX equalization settings of the link partner. The MAC shall drive the relevant 8 GT/s values when the operational rate is 8 GT/s, and it shall drive the relevant 16 GT/s values when the operational rate is 16 GT/s.

7.1.16 Address 408h: PHY TX Control8

This register is used to control transmitter functionality.

<table>
<tr><th>Bit</th><th>Default</th><th>Attribute</th><th>Required</th><th>Description</th></tr>
<tr><td>[7:4]</td><td>0h</td><td>N/A</td><td>N/A</td><td>Reserved</td></tr>
<tr><td>[3]</td><td>0h</td><td>Level</td><td>PCIe</td><td>TxSwing – This field controls transmitter voltage swing level.
<table>
<tr><td>Value</td><td>Description</td></tr>
<tr><td>0</td><td>Full swing</td></tr>
<tr><td>1</td><td>Low swing (optional)</td></tr>
</table>
Implementation of this signal is optional if only Full swing is supported.
This field is not used at the 8.0 GT/s or 16 GT/s signaling rates.</td></tr>
<tr><td>[2:0]</td><td>0h</td><td>Level</td><td>PCIe</td><td>TxMargin[2:0] -- This field selects transmitter voltage levels.
<table>
<tr><td>[2]</td><td>[1]</td><td>[0]</td><td>Description</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>TxMargin value 0 = Normal operating range</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>TxMargin value 1 = 800-1200mV for Full swing* OR
400-700mV for Half swing*</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>TxMargin value 2 = required and vendor defined</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>TxMargin value 3 = required and vendor defined</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>TxMargin value 4 = required and 200-400mV for Full swing* OR 100-200mV for Half swing* if the last value or vendor defined</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>TxMargin value 5 = optional and 200-400mV for Full swing* OR 100-200mV for Half swing* if the last value OR vendor defined OR Reserved if no other values supported</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>TxMargin value 6 = optional and</td></tr>
</table></td></tr>
</table>

							200-400mV for Full swing* OR 100-200mV for Half swing* if the last value OR vendor defined OR Reserved if no other values supported
				1	1	1	TxMargin value 7 = optional and 200-400mV for Full swing* OR 100-200mV for Half swing* if the last value OR Reserved if no other values supported
				PIPE implementations that only support PCI Express mode and the 2.5GT/s signaling rate do not implement this field.			

7.1.17 Address 409h: PHY TX Control9

This register is used to control transmitter functionality.

Bit	Default	Attribute	Required	Description
[7:2]	0h	N/A	N/A	Reserved
[1]	0h	1-cycle	PCIe, USB	**ElasticBufferResetControl –** When asserted, this signal causes the PHY to initiate an EB reset sequence. See section 8.15.3.1 for details. Note: This field is not used in the SerDes architecture
[0]	0h	Level	PCIe, USB	**BlockAlignControl --** This field controls whether the PHY performs block alignment. When BlockAlignControl=0 the PHY disables searching for EIEOS (PCIe)/SYNC OS (USB) on a bit boundary. When BlockAlignControl = 1 the PHY enables searching for EIEOS(PCIe)/SYNC OS (USB) on a bit boundary. A MAC shall set BlockAlignControl to the same value for all active lanes in a link. A MAC shall set BlockAlignControl to '0' when in a datastream and shall set it to '1' otherwise. This field is only used at the PCI Express 8.0 GT/s and 16 GT/s signaling rates and at the USB 10.0 GT/s signaling rate. When the PHY is in Loopback Slave mode it ignores BlockAlignControl and is responsible for maintaining alignment. Note: This field is not used in the SerDes architecture.

7.1.18 Address 800h: PHY Common Control0

This register is used to control functionality relevant to both the receiver and the transmitter functionality.

Bit	Default	Attribute	Required	Description
[7:1]	0h	N/A	N/A	Reserved
[0]	0h	Level	PCIe (optional), USB (optional), SATA	**EncodeDecodeBypass --** This field controls whether the PHY performs 8b/10b (or 128b/13xb) encode and decode. 0 – Encode/decode performed normally by the PHY. 1 – Encode/decode bypassed. The MAC can only change this signal during reset or in a power state other than POWER_STATE_0 (SATA Mode) or P0 (PCI Express Mode). SATA Mode: . When EncodeDecodeBypass is one the TxDataK and RxDataK interfaces are not used and the data bus width is 10, 20, or 40 bits. PCI Express Mode and USB Mode: When EncodeDecodeBypass is one the TxDataK and RxDataK interfaces are not used. The data bus width is 10, 20, or 40 bits if rate is 2.5 or 5.0 GT/s. The data bus width is 8, 16, or 32 bits if the rate is 8.0 GT/s or 16 GT/s (PCI Express) or 10 GT/s (USB). The TxStartBlock and RxStartBlock signals are not used. Note: This field is not used in the SerDes architecture.

7.2 MAC Registers

Table 7-2 lists the MAC registers and their associated address. The details of each register are provided in the subsections below.

Table 7-2 MAC Registers

Byte Address	Register Name	Notes
12'h0	RX Margin Status0	
12'h1	RX Margin Status1	
12'h2	RX Margin Status2	
12'h3	Elastic Buffer Status	Original PIPE only
12'h4	Elastic Buffer Location	Original PIPE only
12'h5	Elastic Buffer Location Update Frequency	Original PIPE only
12'h6	RX Status0	
12'h7	RX Status1	
12'h8	RX Status2	
12'h9	RX Status3	
12'hA	RX Link Evaluation Status0	
12'hB	RX Link Evaluation Status1	
12'hC-12'h3FF	Reserved	
12'h400	TX Status0	
12'h401	TX Status1	
12'h402	TX Status2	
12'h403-12'hFFF	Reserved	

7.2.1 Address 0h: RX Margin Status0

Bit	Default	Attribute	Required	Description
[7:2]	0h	N/A	N/A	Reserved
[1]	0h	1-cycle	PCIe (optional)	**Margin Nak** – This field is used by the PHY to indicate that a voltage margin request corresponds to an unsupported offset that falls within the advertised range. This field may be asserted in response to a change to the 'Start Margin' field or 'Margin Offset[6:0]' field or 'Margin Direction' field during voltage margining only. This field is only written once per committed write affecting either of the above three fields. When this field is set, the 'Margin Status' should not be set. The design must support the minimum voltage offset requirement stated in the PCIe base specification. Note: If the voltage margin offset requested falls outside of the PHY advertised range, the PHY is not required to communicate a NAK by setting this field; this is assumed to be a MAC error and PHY behavior is undefined.

[0]	0h	1-cycle	PCIe	**Margin Status** – This field is used by the PHY to acknowledge a valid change to the 'Start Margin' field or the 'Margin Offset[6:0]' field. This field is only written once per committed write affecting either of the above two fields. For example, if both 'Start Margin' and 'Margin Offset[6:0]' are changed, but one is changed with an uncommitted write and one is changed with a committed write, this 'Margin Status' field is only written once to acknowledge both changes.

7.2.2 Address 1h: RX Margin Status1

Bit	Default	Attribute	Required	Description
[7]	0h	N/A	N/A	Reserved
[6:0]	0h	Level	PCIe (optional)	**Sample Count** – This field indicates the number of bits that have been margined and can increment only when 'Start Margin' is asserted. The value of this field is 3*log2(number of bits margined). This field stops incrementing when the 'Error Count' saturates. This field only resets on a PIPE reset or when the MAC writes to the 'Sample Count Reset' bit in the RX Margin Control1 register. This field is only required if the Sampling Rate is not reported in the PHY datasheet. If used, this field must be updated by the PHY every time the associated value changes; implementations may collapse multiple updates into a single write only to avoid creating a backlog of writes.

7.2.3 Address 2h: RX Margin Status2

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	Level	PCIe (optional)	**Error Count** – This field is only required if errors do not happen in the data stream and thus an independent error sampler is implemented in the PHY. This field is used by the PHY to report actual bit errors to the MAC. This field can increment only when 'Start Margin' is asserted. This field only resets on a PIPE reset or when the MAC writes to the 'Error Count Reset' bit in the RX Margin Control1 register. If used, this field must be updated by the PHY every time the associated value changes; implementations may collapse multiple updates into a single write to avoid creating a backlog of writes.

7.2.4 Address 3h: Elastic Buffer Status

Bit	Default	Attribute	Required	Description
[7:1]	0h	N/A	N/A	Reserved
[0]	0h	1-cycle	PCIe	**Elastic Buffer Status** – The PHY sets this status

			(optional)	bit to 1'b1 when it has completed its elastic buffer depth adjustment to the value specified in the Elastic Buffer Control register. Note: This field is not used in the SerDes architecture.

7.2.5 Address 4h: Elastic Buffer Location

Bit	Default	Attribute	Required	Description
[7:0]	0h	Level	PCIe (optional), USB (optional)	**ElasticBufferLocation --** This field reflects the number of entries currently in the elastic buffer. Whenever the number of entries in the elastic buffer changes the PHY schedules an update to this register, with the frequency of update not to exceed that programmed in the ElasticBufferLocationUpdateFrequency field. Note: This field is not used in the SerDes architecture.

7.2.6 Address 5h: Elastic Buffer Location Update Frequency

Bit	Default	Attribute	Required	Description
[7:0]	5h	Level	No	**ElasticBufferLocationUpdateFrequency --** This field specifies the maximum update frequency to the ElasticBufferLocation field; the frequency of update should not exceed 16*N symbol times, where N is the value programmed in this register. Note: This field is not used in the SerDes architecture.

7.2.7 Address 6h: RX Status0

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	Level	PCIe	**LocalFS[5:0] --** This field reflects the FS value for the PHY. These signals are only used by a PHY that requires dynamic preset coefficient updates. The FS value is valid for 8 GT/s. This field shall be updated by the PHY before PhyStatus deasserts after RESET# and before the first PhyStatus pulse after a rate change to 8 GT/s or in response to GetLocalPresetCoefficients when LocalPresetIndex[4:0] < 11.

7.2.8 Address 7h: RX Status1

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved

[5:0]	0h	Level	PCIe	**LocalLF[5:0] --** This field reflects the LF value for the PHY. This signal is only used by a PHY that requires dynamic preset coefficient updates. The LF value is valid for 8GT/s. LocalLF[5:0] must updated whenever LocalFS[5:0] is updated

7.2.9 Address 8h: RX Status2

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	Level	PCIe	**LocalG4FS[5:0]** This field reflects the FS value for the PHY. These signals are only used by a PHY that requires dynamic preset coefficient updates. The FS value is valid for 16 GT/s. This field shall be updated by the PHY before the first PhyStatus pulse after a rate change to 16 GT/s or in response to GetLocalPresetCoefficients when LocalPresetIndex[4:0] > 10.

7.2.10 Address 9h: RX Status3

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	Level	PCIe	**LocalG4LF[5:0]** This field reflects the LF value for the PHY. This signal is only used by a PHY that requires dynamic preset coefficient updates. The LF value is valid for 16 GT/s. LocalG4LF[5:0] must be sampled whenever LocalG4FS[5:0] is sampled.

7.2.11 Address Ah: RX Link Evaluation Status0

Bit	Default	Attribute	Required	Description
[7:0]	0h	Level	PCIe, Converged IO, DisplayPort RX (optional)	**LinkEvaluationFeedbackFigureMerit[7:0] –** This field provides the PHY link equalization evaluation Figure of Merit value. The value is encoded as an unsigned integer from 0 to 255. An encoding of 0 is the worst, and an encoding of 255 is the best. A PHY does not update this field if it is does not provide link equalization evaluation feedback using the Figure of Merit format. For PCIe, this field is only used at the 8.0 GT/s and 16 GT/s signaling rates. Note: The write_committed associated with an update to this field indicates that the RxEqEval

				has completed.

7.2.12 Address Bh: RX Link Evaluation Status1

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	Level	PCIe	**LinkEvaluationFeedbackDirectionChange[5:0] --** This field provides the link equalization evaluation feedback in the direction change format. Feedback is provided for each coefficient: [1:0] C_{-1} [3:2] C_0 [5:4] C_1 The feedback value for each coefficient is encoded as follows: 00 - No change 01 – Increment 10 – Decrement 11 - Reserved A PHY does not update this field if it is does not provide link equalization evaluation feedback using the Direction Change format. Note: In 8.0 GT/s mode the MAC shall ignore the C_0 value and use the correct value per the PCI Express specification. These signals are only used at the 8.0 GT/s and 16 GT/s signaling rates. Note that C_{-1} and C_1 are encoded as the absolute value of the actual FIR coefficient and thus incrementing or decrementing either value refers to the magnitude of the actual FIR coefficient. For example, if C_{-1} is 000001b the FIR coefficient is negative one and a request to increment C_{-1} will increase it in the direction of 000002b which decreases the FIR coefficient. Note: The write_committed associated with an update to this field indicates that the RxEqEval has completed.

7.2.13 Address 400h: TX Status0

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	1-cycle	PCIe	**LocalTxPresetCoefficients[5:0] --** This field forms

				part of LocalTxPresetCoefficients[17:0], which are the coefficients for the preset on the LocalPresetIndex[4:0] after a GetLocalPresetCoeffcients request: [5:0] C_{-1} [11:6] C_0 [17:12] C_{+1} This field is valid for one PCLK cycle. The MAC will reflect these coefficient values on the TxDeemph bus when MAC wishes to apply this preset. These field is only updated by a PHY that requires dynamic preset coefficient updates

7.2.14 Address 401h: TX Status1

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	1-cycle	PCIe	**LocalTxPresetCoefficients[11:6] --** This field forms part of LocalTxPresetCoefficients[17:0], which are the coefficients for the preset on the LocalPresetIndex[4:0] after a GetLocalPresetCoeffcients request. See LocalTxPresetCoefficients[5:0] description for details.

7.2.15 Address 402h: TX Status2

Bit	Default	Attribute	Required	Description
[7:6]	0h	N/A	N/A	Reserved
[5:0]	0h	1-cycle	PCIe	**LocalTxPresetCoefficients[17:12]] --** This field forms part of LocalTxPresetCoefficients[17:0], which are the coefficients for the preset on the LocalPresetIndex[4:0] after a GetLocalPresetCoeffcients request. See LocalTxPresetCoefficients[5:0] description for details.

8 PIPE Operational Behavior

8.1 Clocking

There are three clock signals used by the PHY Interface component. The first (*CLK*) is a reference clock that the PHY uses to generate internal bit rate clocks for transmitting and receiving data. The specifications for this signal are implementation dependent and must be fully specified by vendors. The specifications may vary for different operating modes of the PHY. This clock may have spread spectrum modulation that matches a system reference clock (for

example, the spread spectrum modulation could come from REFCLK from the Card Electro-Mechanical Specification).

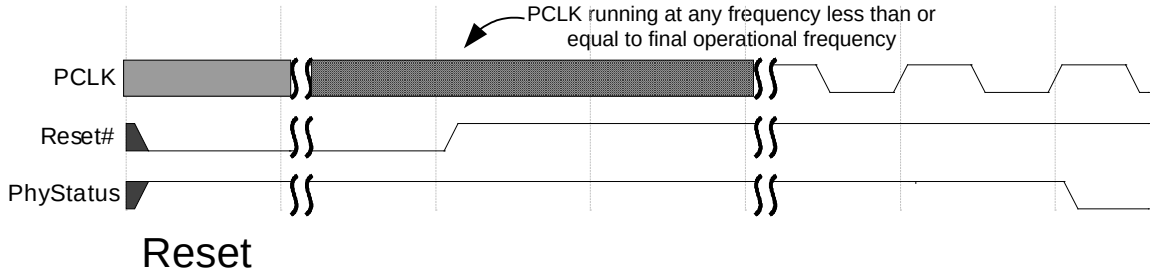
The second clock (*PCLK*) is an output from the PHY in "PCLK as PHY Output" mode and an input to each PHY lane in "PCLK as PHY Input " mode and is the parallel interface clock used to synchronize data transfers across the parallel interface. This clock runs at a rate dependent on the *Rate*, *PCLK Rate*, and *PHY Mode* control inputs and data interface width. The rising edge of this clock is the reference point. This clock may also have spread spectrum modulation. CLK and PCLK must be sourced from the same reference clock and must contain the same clocking characteristics.

The third clock (*MAX PCLK)* is a constant frequency clock with a frequency determined by the maximum signaling rate supported by the PHY and is only required in "PCLK as PHY Input " mode or in all modes for a PHY that supports PCI Express 3.0.

8.2 Reset

When the MAC wants to reset the PHY (e.g.; initial power on), the MAC must hold the PHY in reset until power and *CLK* to the PHY are stable. For PCLK as PHY output, the PHY signals that *PCLK* and/or Max PCLK are valid (i.e. PCLK and/or Max PCLK has been running at its operational frequency for at least one clock) and the PHY is in the specified power state by the deassertion of *PhyStatus* after the MAC has stopped holding the PHY in reset. The MAC must not perform any operational sequences until PhyStatus is returned for the Reset# deassertion. While *Reset#* is asserted the MAC should have *TxDetectRx/Loopback* deasserted, *TxElecIdle* asserted, *TxCompliance* deasserted, *PowerDown* = P1 (PCI Express mode) or *PowerDown* = P2 (USB Mode) or *PowerDown* set to the default value reported by the PHY (SATA Mode), *PHY Mode* set to the desired PHY operating mode, and *Rate* set to 2.5GT/s signaling rate for a PHY in PCI Express mode or 5.0 GT/s or 10 GT/s (highest supported) for a PHY in USB mode or any rate supported by the PHY in SATA mode. The state of *TxSwing* during *Reset#* assertion is implementation specific. *RxTermination* is asserted in USB 3.0 mode.

Figure 8-1. Reset# Deassertion and PhyStatus for PCLK as PHY Output



8.3 Power Management – PCI Express Mode

The power management signals allow the PHY to minimize power consumption. The PHY must meet all timing constraints provided in the PCI Express Base Specification regarding clock recovery and link training for the various power states. The PHY must also meet all terminations

requirements for transmitters and receivers.

Four standard power states are defined, P0, P0s, P1, and P2. P0 state is the normal operational state for the PHY. When directed from P0 to a lower power state, the PHY can immediately take whatever power saving measures are appropriate. A PHY is allowed to implement additional PHY specific power states; L1 substate support requires implementation of additional PHY specific power states. A MAC may use any of the PHY specific states as long as the PCI Express base specification requirements are still met.

In states P0, P0s and P1, PCLK is required to be kept operational. For all state transitions between these three states and any PHY specific states where *PCLK* is operational, the PHY indicates successful transition into the designated power state by a single cycle assertion of *PhyStatus*. Transitions into and out of P2 or a PHY specific state where *PCLK* is not operational are described below. For all power state transitions, the MAC must not begin any operational sequences or further power state transitions until the PHY has indicated that the initial state transition is completed.

Mapping of PHY power states to states in the Link Training and Status State Machine (LTSSM) found in the base specification are included below. A MAC may alternately use PHY specific states as long as the base specification requirements are still met.

- P0 state: All internal clocks in the PHY are operational. P0 is the only state where the PHY transmits and receives PCI Express signaling.
  P0 is the appropriate PHY power management state for most states in the Link Training and Status State Machine (LTSSM). Exceptions are listed below for each lower power PHY state.
- P0s state: *PCLK* must stay operational. The MAC may move the PHY to this state only when the transmit channel is idle.
  P0s state can be used when the transmitter is in state *Tx_L0s.Idle*.

  While the PHY is in either P0 or P0s power states, if the receiver is detecting an electrical idle, the receiver portion of the PHY can take appropriate power saving measures. Note that the PHY must be capable of obtaining bit and symbol lock within the PHY-specified time (N_FTS with/without common clock) upon resumption of signaling on the receive channel. This requirement only applies if the receiver had previously been bit and symbol locked while in P0 or P0s states.
- P1 state: Selected internal clocks in the PHY can be turned off. *PCLK* must stay operational. The MAC will move the PHY to this state only when both transmit and receive channels are idle. The PHY must not indicate successful entry into P1 (by asserting *PhyStatus*) until PCLK is stable and the operating DC common mode voltage is stable and within specification (as per the base spec).
  P1 can be used for the *Disabled* state, all *Detect* states, and *L1.Idle* state (only if L1 substates are not supported) of the Link Training and Status State Machine (LTSSM).
- P2 state: Selected internal clocks in the PHY can be turned off. The parallel interface is in an asynchronous mode and *PCLK* is turned off. P2 can be used for the *L1.Idle*, *L2.Idle* and *L2.TransmitWake* states of the Link Training and Status State Machine (LTSSM).

PCLK as PHY Output: When transitioning into P2, the PHY must assert *PhyStatus* before *PCLK* is turned off and then deassert *PhyStatus* when PCLK is fully off and when the PHY is in the P2 state. When transitioning out of P2, the PHY asserts *PhyStatus* as soon as possible and leaves it

asserted until after *PCLK* is stable.

PCLK as PHY Input: When transitioning into P2, the PHY must assert *PhyStatus* for one input PCLK cycle when it is ready for PCLK to be removed. When transitioning out of P2, the PHY must assert *PhyStatus* for one input PCLK cycle as soon as possible once it has transitioned to P0 and is ready for operation.

When transitioning out of a state that does not provide *PCLK* to another state that does not provide *PCLK*, the PHY asserts *PhyStatus* as soon as the PHY state transition is complete and and leaves it asserted until the MAC asserts *AsyncPowerChangeAck*. Once the MAC asserts *AsyncPowerChangeAck* the PHY deasserts *PhyStatus*.

PHYs should be implemented to minimize power consumption during P2 as this is when the device will have to operate within Vaux power limits (as described in the PCI Express Base Specification).

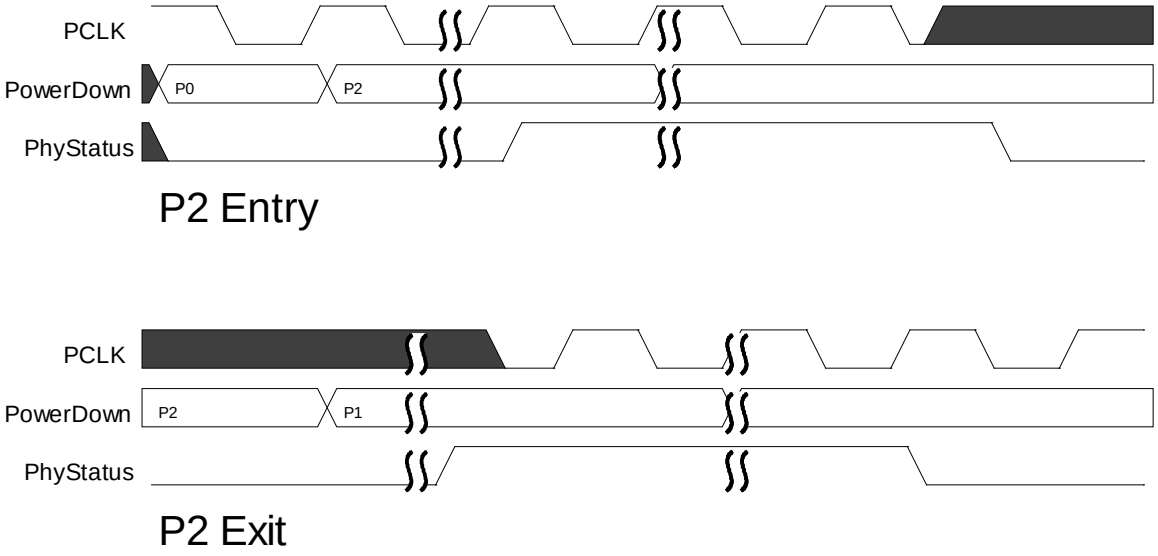


Figure 8-2 PCI Express P2 Entry and Exit with PCLK as PHY Output

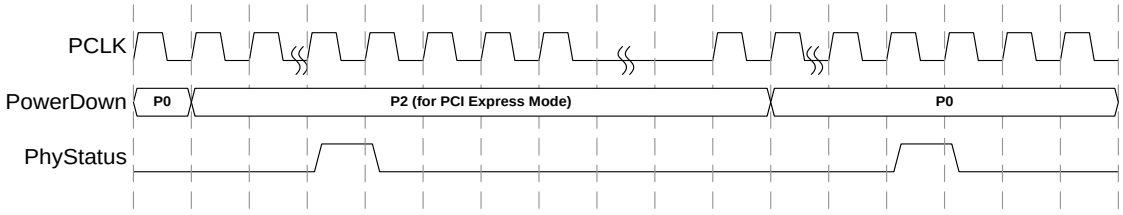


Figure 8-3 PCI Express P2 Entry and Exit with PCLK as PHY Input

There is a limited set of legal power state transitions that a MAC can ask the PHY to make.

Those legal transitions are: P0 to P0s, P0 to P1, P0 to P2, P0s to P0, P1 to P0, and P2 to P0. The base spec also describes what causes those state transitions.

Transitions to and from any pair of PHY power states including at least one PHY specific power state are also allowed by PIPE (unless otherwise prohibited). However, a MAC must ensure that PCI Express specification timing requirements are met.

For L1 substate entry, the PHY must support a state where PCLK is disabled, REFCLK can be removed, and RX electrical idle and TX common mode are on; this can be P2 or a P2-like state. Figure 8-4 illustrates how a transition into and out of an L1 substate could occur. P2 or a P2-like state maps to L1.Idle; and PhyStatus and AsyncPowerChangeAck signals are used as described earlier in this section. Alternatively, the PHY may implement L1 substate management using a single PowerDown[3:0] encoding augmented with the RxEIDetectDisable and TxCommonModeDisable signals; the PowerDown state must remain constant across L1 substate transitions when this alternative mechanism is used. Using distinct PowerDown[3:0] encodings to define the L1 substates allows flexibility to specify different exit latencies; while using RxEIDetectDisable and TxCommonModeDisable may eliminate the need to do a handshake with AsyncPowerChangeAck. The PHY may support either mechanism or both; this capability must be advertised in the PHY datasheet. The sideband mechanism of L1 substate management via RxEIDetectDisable and TxCommonModeDisable requires PCLK as PHY input mode.

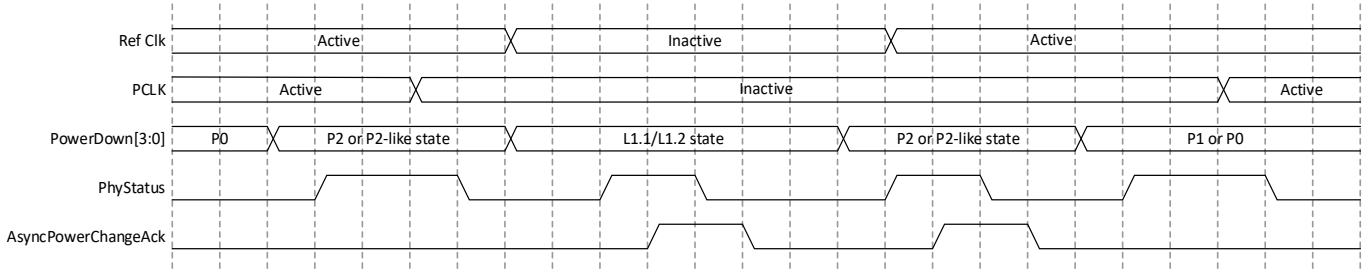


Figure 8-4. L1 SubState Entry and Exit with PCLK as PHY Output

8.4 Power Management – USB Mode

The power management signals allow the PHY to minimize power consumption. The PHY must meet all timing constraints provided in the USB 3.1 Specification regarding clock recovery and link training for the various power states. The PHY must also meet all termination requirements for transmitters and receivers.

Four power states are defined, P0, P1, P2, and P3. The P0 state is the normal operational state for the PHY. When directed from P0 to a lower power state, the PHY can immediately take whatever power saving measures are appropriate.

In states P0, P1 and P2, the PCLK must be kept operational. For all state transitions between these three states, the PHY indicates successful transition into the designated power state by a single cycle assertion of *PhyStatus*. Transitions into and out of P3 are described below. For all power state transitions, the MAC must not begin any operational sequences or further power state transitions until the PHY has indicated that the initial state transition is completed.

Mapping of PHY power states to states in the Link Training and Status State Machine found in the USB specification are included below.

- P0 state: All internal clocks in the PHY are operational. P0 is the only state where the PHY transmits and receives USB signaling.
  P0 is the appropriate PHY power management state for all cases where the link is in U0 and all other link state except those listed below for P1, P2, and P3.
- P1 state: *PCLK* must stay operational. The MAC will move the PHY to this state only when the PHY is transmitting idles and receiving idles. The P1 state can be used for the *U1* link state.
- P2 state: Selected internal clocks in the PHY can be turned off. *PCLK* must stay operational. The MAC will move the PHY to this state only when both transmit and receive channels are idle. The PHY must not indicate successful entry into P2 (by asserting *PhyStatus*) until PCLK is stable and the operating DC common mode voltage is stable and within specification (as per the base spec).
- P2 can be used for the *U2, Rx.Detect, and SS.Inactive*.
- P3 state: Selected internal clocks in the PHY can be turned off. The parallel interface is in an asynchronous mode and *PCLK* output is turned off.

  PCLK as PHY Output: When transitioning into P3, the PHY must assert *PhyStatus* before *PCLK* is turned off and then deassert *PhyStatus* when PCLK is fully off and when the PHY is in the P3 state. When transitioning out of P3, the PHY asserts *PhyStatus* as soon as possible and leaves it asserted until after *PCLK* is stable.

  PCLK as PHY Input: When transitioning into P3, the PHY must assert *PhyStatus* for one input PCLK cycle when it is ready for PCLK to be removed. When transitioning out of P3, the PHY must assert *PhyStatus* for one input PCLK cycle as soon as possible once it has transitioned to P0 and is ready for operation.

PHYs should be implemented to minimize power consumption during P3 as this is when the device will have to operate within power limits described in the USB 3.0 Specification.

- The P3 state shall be used in states *SS.disabled and U3.*
- There is a limited set of legal power state transitions that a MAC can ask the PHY to make. Referencing the main state diagram in the USB spec and the mapping of link states to PHY power states described in the preceding paragraphs, those legal transitions are: P0 to P1, P0 to P2, P0 to P3, P1 to P0, P2 to P0, P3 to P0, and P1 to P2. The base spec also describes what causes those state transitions.

8.5 Power Management – SATA Mode

The power management signals allow the PHY to minimize power consumption. The PHY must meet all timing constraints provided in the SATA Specification regarding clock recovery and link training for the various power states. The PHY must also meet all termination requirements for transmitters and receivers.

A minimum of five power states are defined, POWER_STATE_0 and a minimum of four additional states that meet minimum requirements defined in section 6.1. POWER_STATE_0 state is the normal operational state for the PHY. When directed from POWER_STATE_0 to a lower power state, the PHY can immediately take whatever power saving measures are appropriate.

For all state transitions between POWER_STATE_0 and lower power states that provide PCLK, the PHY indicates successful transition into the designated power state by a single cycle assertion of *PhyStatus*. The PHY must complete transmitting all data transferred across the PIPE interface before the change in the PowerDown signals before assertion of *PhyStatus.* Transitions into and out of power states that do not provide PCLK are described below. For all power state transitions, the MAC must not begin any operational sequences or further power state transitions until the PHY has indicated that the initial state transition is completed. Power state transitions between two power states that do not provide PCLK are not allowed.

Mapping of PHY power states to link states in the SATA specification is MAC specific.

- POWER_STATE_0 : All internal clocks in the PHY are operational. POWER_STATE_0 is the only state where the PHY transmits and receives SATA signaling.
  POWER_STATE_0 is the appropriate PHY power management state for most link states in the SATA specification. When transitioning into a power state that does not provide *PCLK* , the PHY must assert *PhyStatus* before *PCLK* is turned off and then deassert *PhyStatus* when PCLK is fully off and when the PHY is in the low power state. The PHY must leave PCLK on for at least one cycle after asserting *PhyStatus.* For PCLK as PHY output, when transitioning out of a state that does not provide *PCLK* , the PHY asserts *PhyStatus* as soon as possible and leaves it asserted until after *PCLK* is stable.

Transitions between any pair of PHY power states (except two states that do not provide PCLK) are allowed by PIPE. However, a MAC must ensure that SATA specification timing requirements are met.

8.6 Changing Signaling Rate, PCLK Rate, or Data Bus Width

8.6.1 PCI Express Mode

The signaling rate of the link, PCLK rate, or the Data Bus Width can be changed only when the PHY is in the P0 or P1 power state and *TxElecIdle* and *RxStandby* (P0 only) are asserted. When the MAC changes the *Rate* signal, and/or the *Width* signal, and/or the *PCLK rate* signal in PCLK as PHY Output mode, the PHY performs the rate change and/or the width change and/or the PCLK rate change and signals its completion with a single cycle assertion of *PhyStatus*. The MAC must not perform any operational sequences, power state transitions, deassert *TxElecIdle* or *RxStandby*, or further signaling rate changes until the PHY has indicated that the signaling rate change has completed. The sequence is the same in PCLK as PHY Input mode except the MAC needs to know when the input PCLK rate can be safely changed. After the MAC changes PCLK_Rate the change to the PCLK can happen only after the PclkChangeOk output has been driven high by the PHY. The MAC changes the input PCLK, and then handshakes by asserting PclkChangeAck. The PHY responds by asserting PhyStatus for one input PCLK cycle and de-asserts PclkChangeOk on the trailing edge of PhyStatus. Note: PclkChangeOk is only used by the PHY if the MAC changes PCLK_Rate. The MAC de-asserts PclkChangeAck when PclkChangeOk is sampled low and may de-assert *TxElecIdle* and/or *RxStandby* after PhyStatus is sampled high. There are instances where LTSSM state machine transitions indicate both a speed change and/or width and/or PCLK rate change and a power state change for the PHY. In these instances, the MAC must change (if necessary) the signaling rate, width and/or PCLK rate before changing the power state.

Some PHY architectures may allow a speed change and a power state change to occur at the same time as a rate and/or width and/or PCLK rate change. If a PHY supports this, the MAC must

change the rate and/or width and/or PCLK rate at the same PCLK edge that it changes the *PowerDown* signals. This can happen when transitioning the PHY from P0 to either P1 or P2 states. The completion mechanisms are the same as previously defined for the power state changes and indicate not only that the power state change is complete, but also that the rate and/or width and/or PCLK rate change is complete.

8.6.2 USB Mode

The signaling rate of the link, PCLK rate, or the Data Bus Width can be changed only when the PHY is in the P0 or P2 power state and *TxElecIdle* and *RxStandby are* asserted. Any combination of at least two of the rate and width and PCLK rate, can be changed simultaneously. The MAC is not allowed to change only one of the three. When the MAC changes the *Rate* signal, and/or the *Width* signal, and/or the *PCLK rate* signal in PCLK as PHY Output mode, the PHY performs the rate change and/or the width change and/or the PCLK rate change and signals its completion with a single cycle assertion of *PhyStatus*. The MAC must not perform any operational sequences, power state transitions, deassert *TxElecIdle* or *RxStandby*, or further signaling rate changes until the PHY has indicated that the signaling rate change has completed. The sequence is the same in PCLK as PHY Input mode except the MAC needs to know when the input PCLK rate can be safely changed. After the MAC changes PCLK_Rate the change to the PCLK can happen only after the PclkChangeOk output has been driven high by the PHY. The MAC changes the input PCLK, and then handshakes by asserting PclkChangeAck. The PHY responds by asserting PhyStatus for one input PCLK cycle and de-asserts PclkChangeOk on the trailing edge of PhyStatus. Note: PclkChangeOk is only used by the PHY if the MAC changes PCLK_Rate. The MAC de-asserts PclkChangeAck when PclkChangeOk is sampled low and may de-assert *TxElecIdle* and/or *RxStandby* after PhyStatus is sampled high.

Some PHY architectures may allow a speed change and a power state change to occur at the same time as a rate and/or width and/or PCLK rate change. If a PHY supports this, the MAC must change the rate and/or width and/or PCLK rate at the same PCLK edge that it changes the *PowerDown* signals. This can happen when transitioning the PHY from P0 to either P2 or P3 states. The completion mechanisms are the same as previously defined for the power state changes and indicate not only that the power state change is complete, but also that the rate and/or width and/or PCLK rate change is complete.

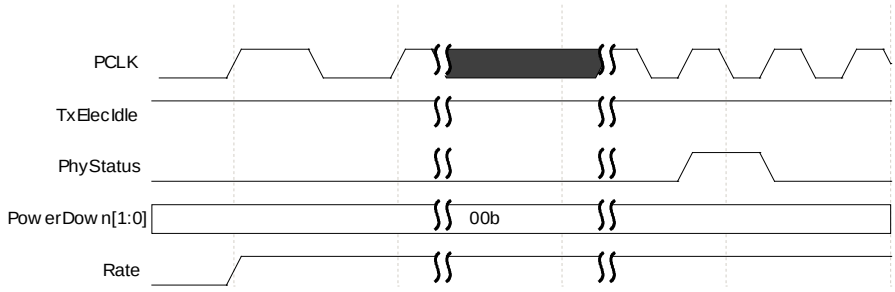
8.6.3 SATA Mode

The signaling rate of the link, PCLK rate, or the Data Bus Width can be changed only when the PHY is in POWER_STATE_0 and *TxElecIdle* and *RxStandby are asserted,* or in a lowpower state where PCLK is provided. When the MAC changes the *Rate* signal, and/or the *Width* signal, and/or the *PCLK rate* signal in PCLK as PHY Output mode, the PHY performs the rate change and/or the width change and/or the PCLK rate change and signals its completion with a single cycle assertion of *PhyStatus*. The MAC must not perform any operational sequences, power state transitions, deassert *TxElecIdle* or *RxStandby*, or further signaling rate and/or width changes until the PHY has indicated that the change has completed.
The sequence is the same in PCLK as PHY Input mode except the MAC needs to know when the input PCLK rate can be safely changed. After the MAC changes PCLK_Rate the change to the PCLK can happen only after the PclkChangeOk output has been driven high by the PHY. The MAC changes the input PCLK, and then handshakes by asserting PclkChangeAck. The PHY responds by asserting PhyStatus for one input PCLK cycle and de-asserts PclkChangeOk on the trailing edge of PhyStatus. Note: PclkChangeOk is only used by the PHY if the MAC changes

PCLK_Rate. The MAC de-asserts PclkChangeAck when PclkChangeOk is sampled low and may de-assert *TxElecIdle* and/or *RxStandby* after PhyStatus is sampled high.
There are instances where conditions indicate both a speed change and/or width and/or PCLK rate change and a power state change for the PHY. In such cases the MAC must change the signaling rate and/or width and/or PCLK rate, before changing the power state.
Some PHY architectures may allow a speed change and a power state change to occur at the same time as a rate and/or width and/or PCLK rate change. If a PHY supports this, the MAC must change the rate and/or width and/or PCLK rate at the same PCLK edge that it changes the *PowerDown* signals. The completion mechanisms are the same as previously defined for the power state changes and indicate not only that the power state change is complete, but also that the rate and/or width and/or PCLK rate change is complete.

8.6.4 Fixed data path implementations

The figure below shows logical timings for implementations that change PCLK frequency when the MAC changes the signaling rate and PCLK is a PHY Output. Implementations that change the *PCLK* frequency when changing signaling rates must change the clock such that the time the clock is stopped (if it is stopped) is minimized to prevent any timers using *PCLK* from exceeding their specifications. Also during the clock transition period, the frequency of *PCLK* must not exceed the PHY's defined maximum clock frequency. The amount of time between when *Rate* is changed and the PHY completes the rate change is a PHY specific value. These timings also apply to implementations that keep the data path fixed by using options that make use of the TxDataValid and RxDataValid signals.



Rate change with fixed data path

Figure 8-5 shows logical timings for an implementation that changes PCLK frequency when the MAC changes the signaling rate and PCLK is a PHY Input.

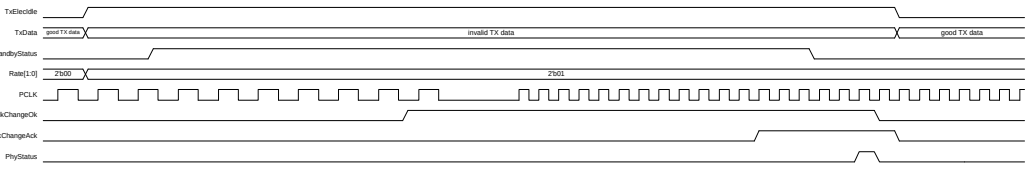
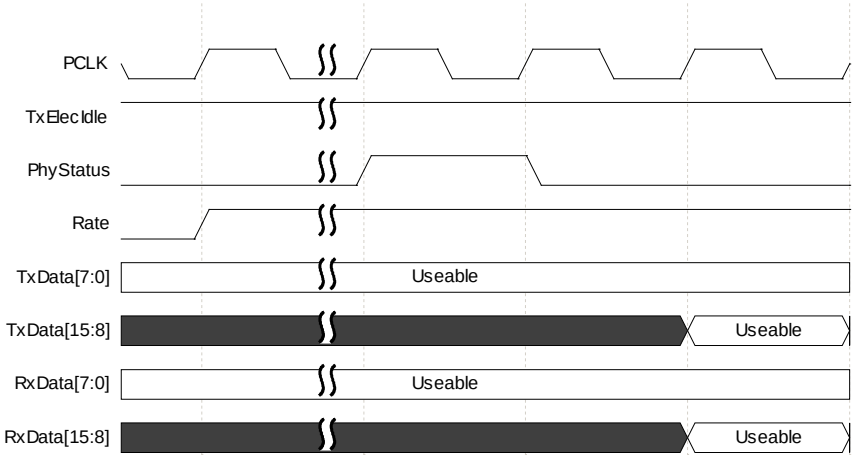


Figure 8-5 Change from PCI Express 2.5 Gt/s to 5.0 Gt/s with PCLK as PHY Input.

8.6.5 Fixed PCLK implementations

The figure below shows logical timings for implementations that change the width of the data path for different signaling rates. PCLK may be stopped during a rate change. These timings

also apply to fixed PCLK implementations that make use of the TxDataValid and RxDataValid signals.

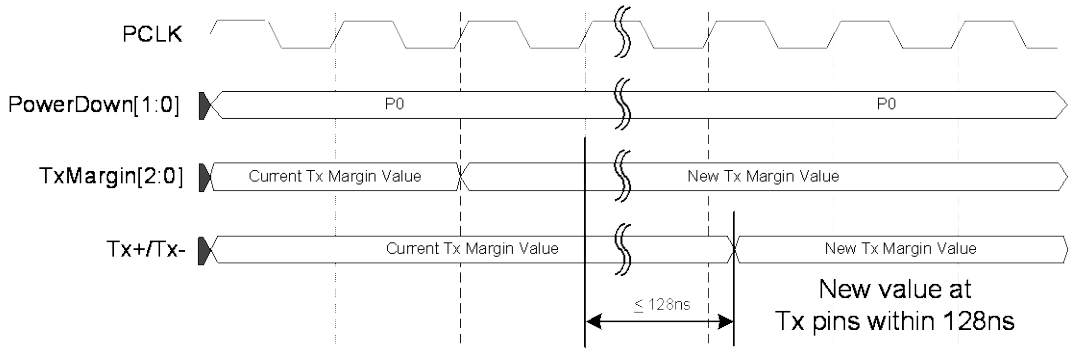


Rate change with fixed PCLK frequency

8.7 Transmitter Margining – PCI Express Mode and USB Mode

While in the P0 power state, the PHY can be instructed to change the value of the voltage at the transmitter pins. When the MAC changes *TxMargin[2:0],* the PHY must be capable of transmitting with the new setting within 128 ns.

There is a limited set of legal *TxMargin[2:0]* and *Rate* combinations that a MAC can select. Refer to the PCIe Base Specification for a complete description of legal settings when the PHY is in PCI Express Mode. Refer to the USB specification for a complete description of the legal settings when the PHY is in USB mode.



Selecting Tx Margining value

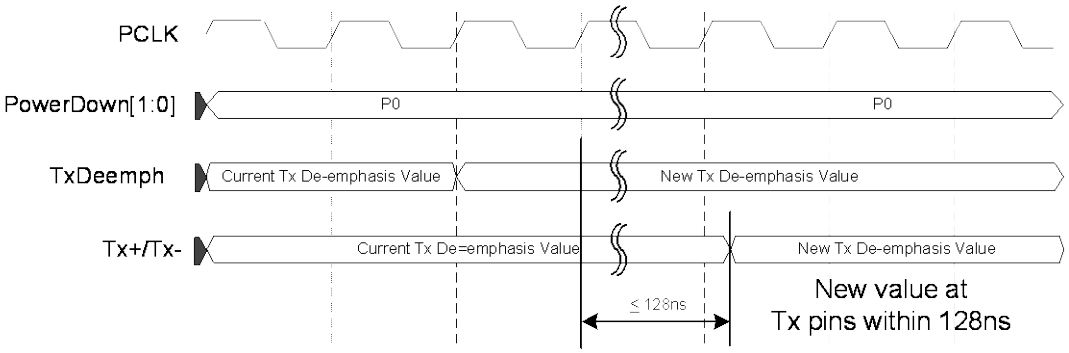
8.8 Selectable De-emphasis – PCI Express Mode

While in the P0 power state and transmitting at 5.0GT/s, 8.0 GT/s or 16 GT/s, the PHY can be instructed to change the value of the transmitter equalization. When the signaling rate is 5.0 GT/s and the MAC changes *TxDeemph,* the PHY must be capable of transmitting with the new setting

within 128 ns. When the signaling rate is 8.0 GT/s or 16 GT/s and the MAC changes *TxDeemph*, the PHY must be capable of transmitting with the new setting within 256 ns.

There is a limited set of legal *TxDeemph* and *Rate* combinations that a MAC can select. Refer to the PCIe Base Specification for a complete description.

The MAC must ensure that *TxDeemph* is selecting -3.5db whenever *Rate* is selecting 2.5 GT/s.

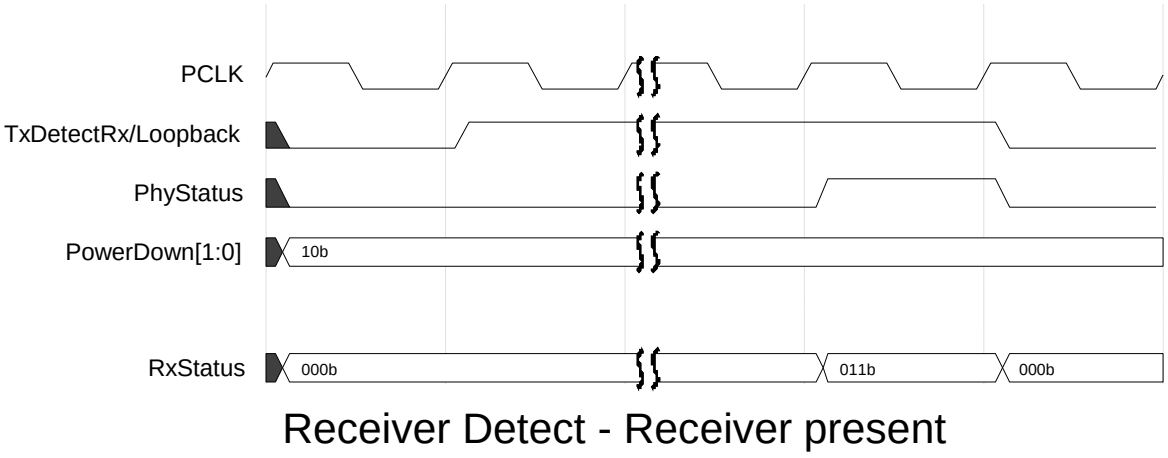


Selecting Tx De-emphasis value

8.9 Receiver Detection – PCI Express Mode and USB Mode

While in the P1 power state and PCI Express mode or in the P2 or P3 power state and USB mode, the PHY can be instructed to perform a receiver detection operation to determine if there is a receiver at the other end of the link. Basic operation of receiver detection is that the MAC requests the PHY to do a receiver detect sequence by asserting *TxDetectRx/Loopback*. When the PHY has completed the receiver detect sequence, it asserts *PhyStatus* for one clock and drives the *RxStatus* signals to the appropriate code. After the receiver detection has completed (as signaled by the assertion of *PhyStatus*), the MAC must deassert *TxDetectRx/Loopback* before initiating another receiver detection, a power state transition, or signaling a rate change.
Once the MAC has requested a receiver detect sequence (by asserting *TxDetectRx/Loopback*), the MAC must leave *TxDetectRx/Loopback* asserted until after the PHY has signaled completion by the assertion of *PhyStatus*. When receiver detection is performed in USB mode with the PHY in P3 the PHY asserts PhyStatus and signals the appropriate receiver detect value until the MAC deasserts TxDetectRx/Loopback.

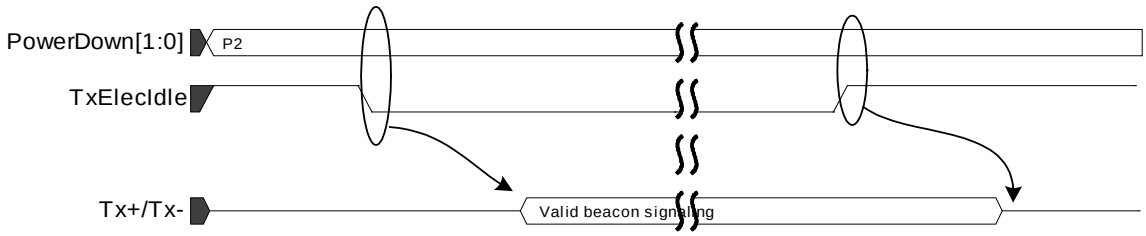
Detected Condition	*RxStatus* code
Receiver not present	000b
Receiver present	011b



Receiver Detect - Receiver present

8.10 Transmitting a beacon – PCI Express Mode

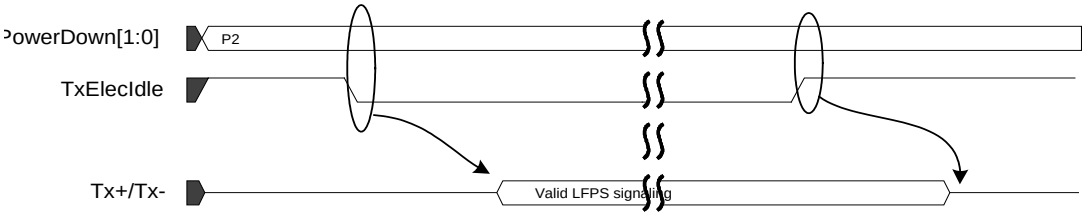
When the PHY has been put in the P2 power state, and the MAC wants to transmit a beacon, the MAC deasserts *TxElecIdle* and the PHY should generate a valid beacon until *TxElecIdle* is asserted. The MAC must assert *TxElecIdle* before transitioning the PHY to P0.



Beacon Transmit

8.11 Transmitting LFPS – USB Mode

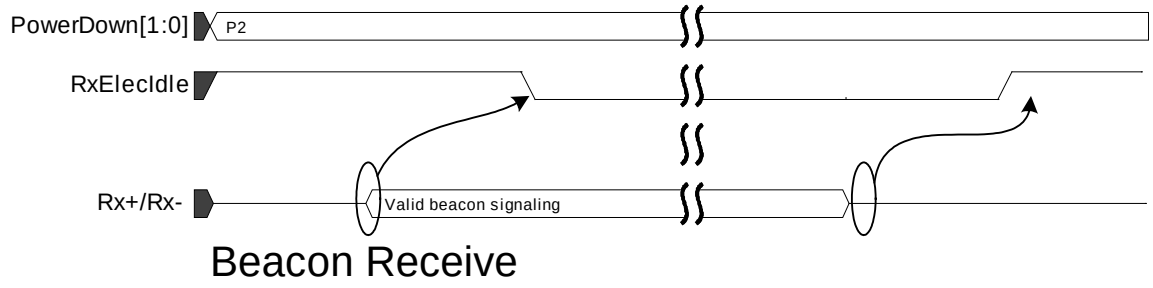
When the PHY is in P1, P2, or P3 and the MAC wants to transmit LFPS, the MAC deasserts *TxElecIdle* and the PHY should generate valid LFPS until *TxElecIdle* is asserted. The MAC must assert *TxElecIdle* before transitioning the PHY to P0. The length of time *TxElecIdle* is deasserted is varied for different events. When the PHY is in P0 and the MAC wants to transmit LFPS, the MAC must assert both *TxElecIdle* and *TxDetectRx/Loopback* for the desired duration of an LFPS burst. The PHY is required to complete a full LFPS period before transitioning to SuperSpeed data, and as a consequence may drop SuperSpeed data if these requests overlap. This requirement does not apply to TxOnesZeros requests. Refer to chapter 6 in the USB 3.0 specification for more details.



LFPS Transmit

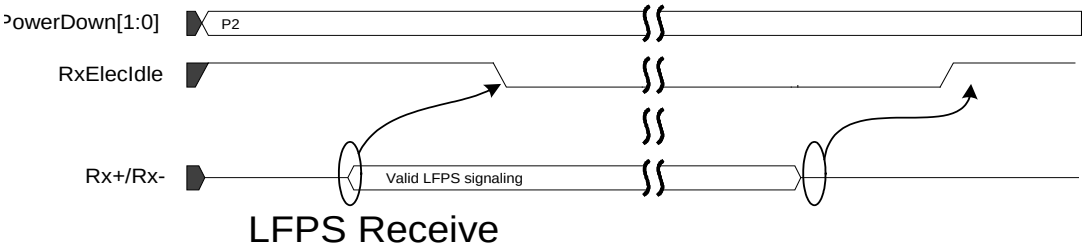
8.12 Detecting a beacon – PCI Express Mode

The PHY receiver must monitor at all times (except during reset or when RxEIDetectDisable is set) for electrical idle. When the PHY is in the P2 power state, and *RxElecIdle* is deasserted, then a beacon is being detected.



8.13 Detecting Low Frequency Periodic Signaling – USB Mode

The PHY receiver must monitor at all times (except during reset, when RX terminations are removed, or when RxEIDetectDisable is set) for LFPS. When the PHY is in the P0, P1, P2, or P3 power state, and *RxElecIdle* is deasserted, then LFPS is being detected. The length of time *RxElecIdle* is deasserted indicates the length of time Low Frequency Periodic Signaling is detected. Refer to chapter 6 in the USB 3.0 specification for more details on the length of Low Frequency Periodic Signaling (LFPS) for various events.



8.14 Clock Tolerance Compensation

The PHY receiver contains an elastic buffer used to compensate for differences in frequencies between bit rates at the two ends of a Link. The elastic buffer must be capable of holding enough symbols to handle worst case differences in frequency and worst case intervals between symbols that can be used for rate compensation for the selected PHY mode.

Two models are defined for the elastic buffer operation in the PHY. The PHY may support one or both of these models. The Nominal Empty buffer model is only supported in PCI Express, USB or SATA Mode.

For the Nominal Empty buffer model the PHY attempts to keep the elasticity buffer as close to empty as possible. In Nominal Empty mode the PHY uses the RxDataValid interface to tell the MAC when no data is available. The Nominal Empty buffer model provides a smaller worst case and average latency then the Nominal Half Full buffer model, but requires the MAC to support the RxDataValid signal. The PHY removes all SKP symbols in Nominal Empty buffer mode.

For the Nominal Half Full buffer model, the PHY is responsible for inserting or removing SKP symbols, ordered sets, or ALIGNs in the received data stream to avoid elastic buffer overflow or underflow. The PHY monitors the receive data stream, and when a Skip ordered-set or ALIGN is received, the PHY can add or remove one SKP symbol (PCI Express Mode at 2.5 or 5 GT/s) or four SKP symbols (PCI Express Mode at 8 GT/s or 16 GT/s) or one SKP ordered set (USB Mode at 5 GT/s) or one ALIGN from each SKP or ALIGN as appropriate to manage its elastic buffer to keep the buffer as close to half full as possible. In USBmode at 5 GT/S the PHY shall only add or remove SKP ordered sets. In USB mode at 10 GT/s the PHY shall only add or remove multiples of four SKP symbols. Whenever SKP symbol(s) or an ordered set is added to or removed, the PHY will signal this to the MAC using the *RxStatus[2:0]* signals. These signals have a non-zero value for one clock cycle and indicate whether a SKP symbol or ordered set was added to or removed from the received SKP ordered-set(s). For PCI Express, the timing of *RxStatus[2:0]* assertion depends on the operational rate since SKP ordered sets are encoded differently in 8b/10b mode versus 128/130b mode. In PCI Express Mode at 2.5 or 5 GT/s, *RxStatus[2:0]* shall be asserted during the clock cycle when the COM symbol of the SKP ordered-set is moved across the parallel interface. In PCI Express Mode at 8 or 16 GT/s, *RxStatus[2:0]* shall assert anytime between and including the start of the SKP ordered set and the SKP_END symbol. In SATA Mode whenever a ALIGN symbol is added or removed, the PHY will signal this to the MAC using the *RxStatus[2:0]* signals. These signals have a non-zero value for one clock cycle and indicate whether an ALIGN was added or removed. *RxStatus* shall be asserted during the clock cycle when the first symbol of the added ALIGN is moved across the parallel interface.

In PCI Express mode, the rules for operating in Nominal Empty buffer mode are as follows:

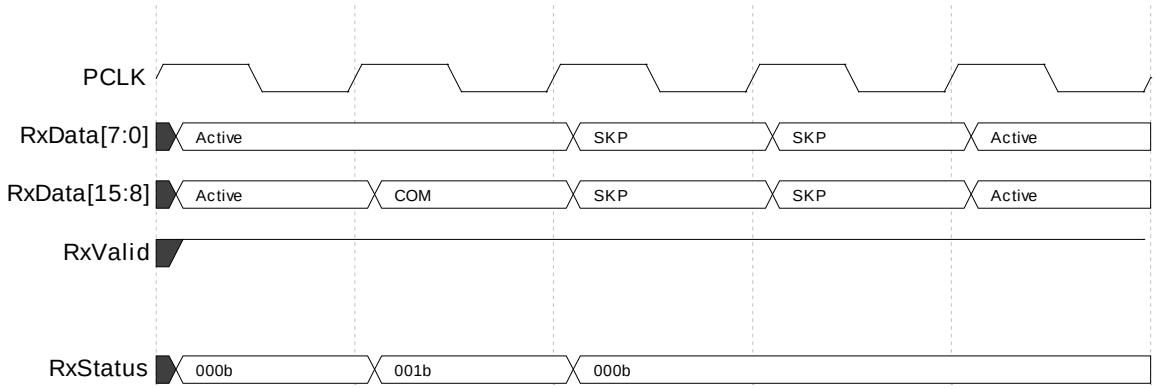
- Use of RxDataValid is required
- All SKP symbols of SOS are removed (8b/10b SKP or 128/130 AA)
- When an empty condition happens (caused by clock drift or SOS removal)
    - RxValid must remain high
        - RxValid should only be dropped for symbol alignment loss or block alignment loss
    - RxDataValid must be de-asserted
    - RxStatus must be 0
- EB full can still occur and is considered an error
- Notification of an SOS coming through the EB must be reported in the following manner
    - 8b/10b: COM of SOS must be passed with RxStatus = SKP removed (010), SKP symbols dropped
    - 128/130: Start of SOS block, with first byte SKP_END or SKP_END_CTRL, must be passed with RxStatus = SKP Removed (010), all AA SKP symbols dropped
- The EB is permitted to start RxDataValid as soon as data is available, but should never assert faster than the usual RxDataValid rate
    - i.e. rate=1, width=2, pclk_rate=2, RxDataValid should never assert for two consecutive pclk cycles
    - i.e. rate=1, width=2, pclk_rate=3, RxDataValid assertions must always have at least 3 pclk cycles of de-assertion between them
        - Example of valid optimization by EB:
            - Rate=1, width=2, pclk_rate=3
            - RxDataValid (t=0,t=1, etc., E=EB Empty):
                - 1000100010001000EE100010001
            - Vs. non-optimized:

- 10001000100010000EE00100010001
- Non-optimized design builds EB depth in-order to maintain RxDataValid fixed cycle rate

In USB mode for the Nominal Empty buffer model the PHY attempts to keep the elasticity buffer as close to empty as possible. This means that the PHY will be required to insert SKP ordered sets into the received data stream when no SKP ordered sets have been received, unless the RxDataValid signal is used. The Nominal Empty buffer model provides a smaller worst case and average latency then the Nominal Half Full buffer model, but requires the MAC to support receiving SKP ordered sets any point in the data stream.

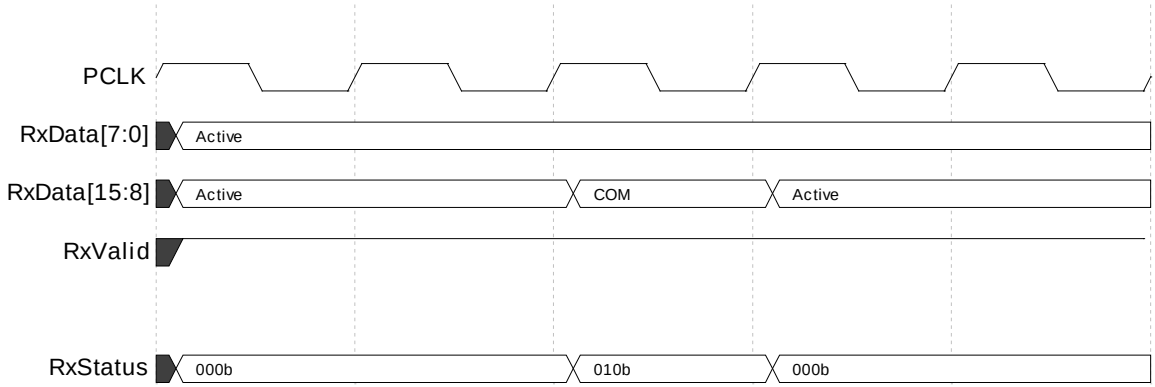
In SATA mode for the Nominal Empty buffer model the PHY attempts to keep the elasticity buffer as close to empty as possible. In Nominal Empty mode the PHY uses the RxDataValid interface to tell the MAC when no data is available. The Nominal Empty buffer model provides a smaller worst case and average latency then the Nominal Half Full buffer model, but requires the MAC to support the RxDataValid signal.

It is recommended that a PHY and MAC support the Nominal Empty buffer model in USB mode using the RxDataValid signal. The alternative of inserting SKPs in the data stream when no SKPs have been received is not recommended. The figure below shows a sequence where a PHY operating in PCI Express Mode added a SKP symbol in the data stream.



Clock Correction - Add a SKP

The figure below shows a sequence where a PHY operating in PCI Express mode removed a SKP symbol from a SKP ordered-set that only had one SKP symbol, resulting in a ‘bare’ COM transferring across the parallel interface.



Clock Correction - Remove a SKP

8.15 Error Detection

The PHY is responsible for detecting receive errors of several types. These errors are signaled to the MAC layer using the receiver status signals (*RxStatus[2:0]*). Because of higher level error detection mechanisms (like CRC) built into the Data Link layer there is no need to specifically identify symbols with errors, but reasonable timing information about when the error occurred in the data stream is important. When a receive error occurs, the appropriate error code is asserted for one clock cycle at the point in the data stream across the parallel interface closest to where the error actually occurred. There are four error conditions (five for SATA mode) that can be encoded on the *RxStatus* signals. If more than one error should happen to occur on a received byte (or set of bytes transferred across a 16-bit, 32-bit or 64-bit interface), the errors should be signaled with the priority shown below.

1. 8B/10B decode error or block decode error
2. Elastic buffer overflow
3. Elastic buffer underflow (Cannot occur in Nominal Empty buffer model)
4. Disparity errors
5. Misalign (SATA mode only)

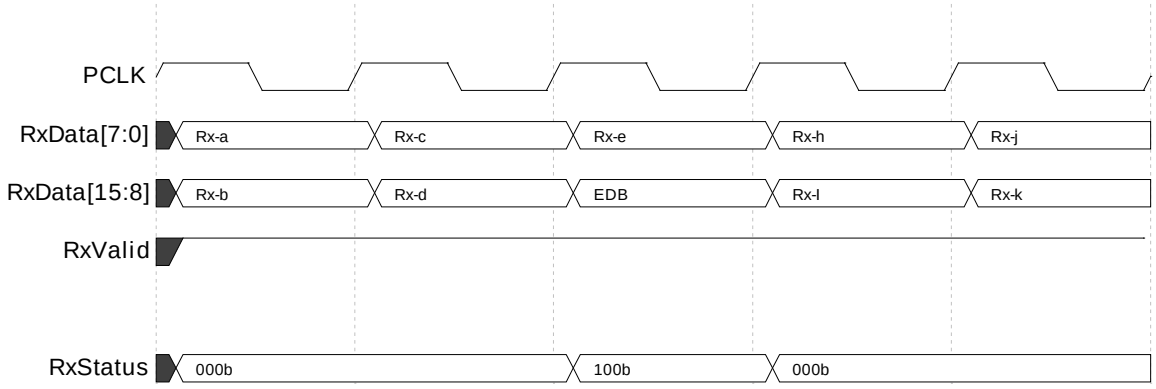
If an error occurs during a SKP ordered-set or ALIGN, such that the error signaling and SKP or ALIGN added/removed signaling on *RxStatus* would occur on the same CLK, then the error signaling has precedence.

Note that the PHY does not signal 128/130B (PCI Express) or 128/132B (USB) header errors. The raw received header bits are passed across the interface and the controller is responsible for any block header error detection/handling.

8.15.1 8B/10B Decode Errors

For a detected 8B/10B decode error, the PHY should place an EDB symbol (for PCIe or SATA) or SUB symbol (for USB) in the data stream in place of the bad byte, and encode *RxStatus* with a decode error during the clock cycle when the effected byte is transferred across the parallel interface. In the example below, the receiver is receiving a stream of bytes Rx-a through Rx-z, and byte Rx-f has an 8B/10B decode error. In place of that byte, the PHY places an EDB (for PCIe or SATA) or SUB (for USB) on the parallel interface, and sets *RxStatus* to the 8B/10B decode error code. Note that a byte that can't be decoded may also have bad disparity, but the 8B/10B error has precedence. Also note that for greater than 8-bit interface, if the bad byte is on the lower byte lane, one of the other bytes may have bad disparity, but again, the 8B/10B error has

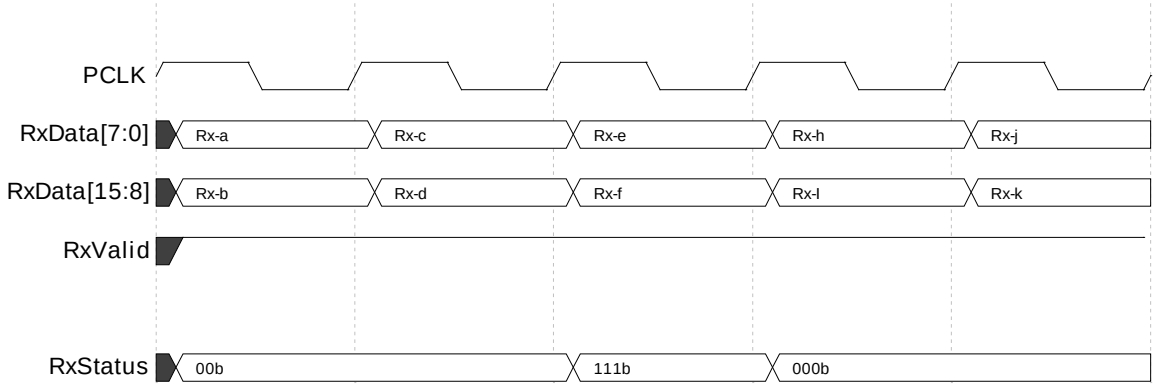
precedence.



8B/10B Decode Error

8.15.2 Disparity Errors

For a detected disparity error, the PHY should assert *RxStatus* with the disparity error code during the clock cycle when the affected byte is transferred across the parallel interface. For greater than 8-bit interfaces, it is not possible to discern which byte (or possibly both) had the disparity error. In the example below, the receiver detected a disparity error on either (or both) Rx-e or Rx-f data bytes, and indicates this with the assertion of *RxStatus*. Optionally, the PHY can signal disparity errors as 8B/10B decode error (using code 0b100). (MACs often treat 8B/10B errors and disparity errors identically.). When operating in USBmode signaling disparity errors is optional.



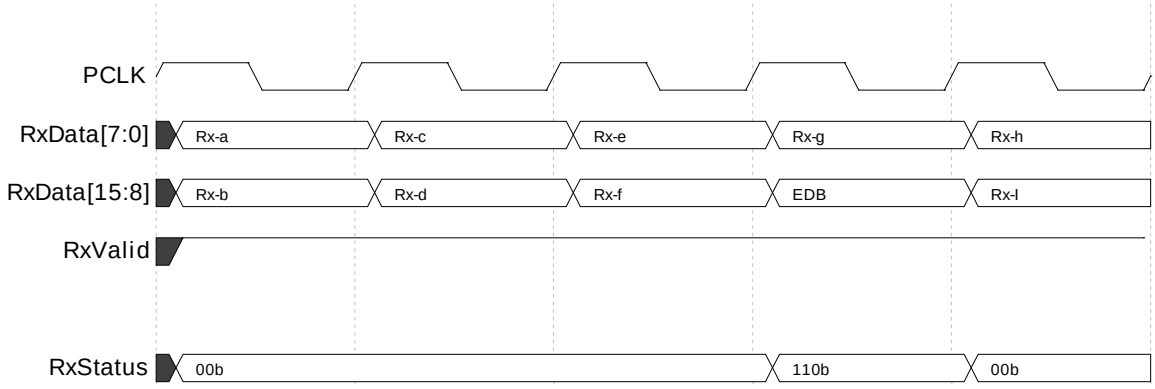
Disparity Error

8.15.3 Elastic Buffer Errors

For elastic buffer errors, an underflow should be signaled during the clock cycle or clock cycles when a spurious symbol is moved across the parallel interface. The symbol moved across the interface should be the EDB symbol (for PCIe or SATA) or SUB symbol (for USB). In the timing diagram below, the PHY is receiving a repeating set of symbols Rx-a thru Rx-z. The elastic buffer underflows causing the EDB symbol (for PCIe) or SUB symbol (for USB) to be inserted between the Rx-g and Rx-h Symbols. The PHY drives *RxStatus* to indicate buffer underflow during the clock cycle when the EDB (for PCIe) or SUB (for USB) is presented on the parallel interface.

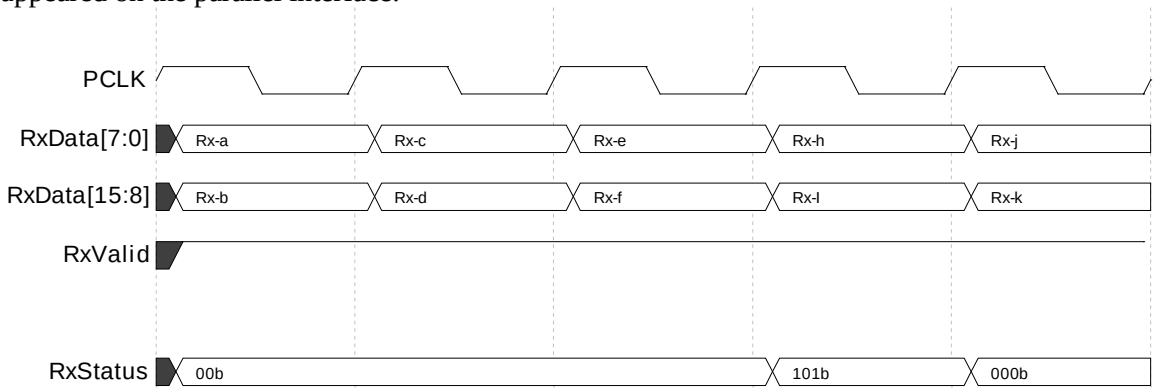
Note that underflow is not signaled when the PHY is operating in Nominal Empty buffer mode.

In this mode SKP ordered sets are moved across the interface whenever data needs to be inserted or the RxDataValid signal is used. The RxDataValid method is preferred.



Elastic Buffer Underflow

For an elastic buffer overflow, the overflow should be signaled during the clock cycle where the dropped symbol or symbols would have appeared in the data stream. For the 16-bit interface it is not possible, or necessary, for the MAC to determine exactly where in the data stream the symbol was dropped. In the timing diagram below, the PHY is receiving a repeating set of symbols Rx-a thru Rx-z. The elastic buffer overflows causing the symbol Rx-g to be discarded. The PHY drives *RxStatus* to indicate buffer overflow during the clock cycle when Rx-g would have appeared on the parallel interface.



Elastic Buffer Overflow

8.15.3.1 Elastic Buffer Reset

The MAC can set the ElasticBufferResetControl bit (see section 7.1.17) to initiate an EB reset sequence in the PHY. The PHY must complete the EB reset sequence within 16 PCLK cycles as follows:

- Assert RxStatus to value of 1xx with RxValid
- Hold RxStatus to 1xx while maintaining RxValid and RxDataValid
- Move pointers back to their initial state
- Release RxStatus to indicate clean data is being forwarded again

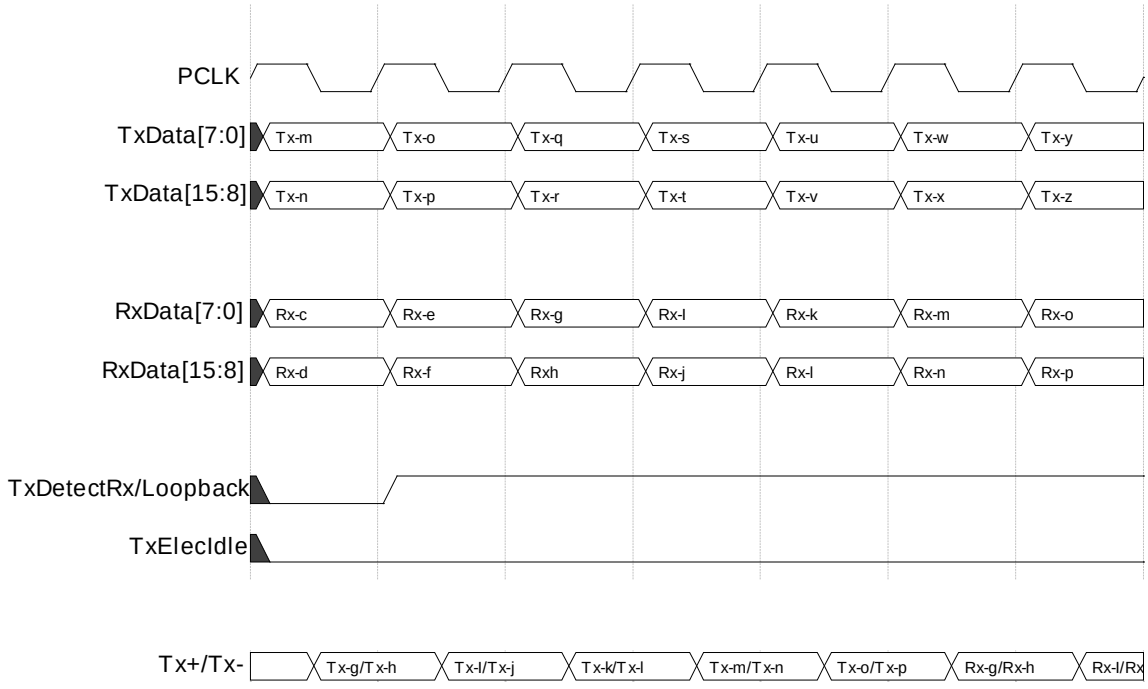
8.16 Loopback

- For USB and PCI Express Modes the PHY must support an internal loopback as described in the corresponding base specification.
- For SATA the PHY may optionally support an internal loopback mode when EncodeDecodeBypass is asserted.
- In the SerDes architecture, loopback is handled in the MAC instead of the PHY.

The PHY begins to loopback data when the MAC asserts *TxDetectRx/Loopback* while doing normal data transmission (i.e. when *TxElecIdle* is deasserted). The PHY must, within the specified receive and transmit latencies, stop transmitting data from the parallel interface, and begin to loopback received symbols. While doing loopback, the PHY continues to present received data on the parallel interface.

The PHY stops looping back received data when the MAC deasserts *TxDetectRx/Loopback*. Transmission of data on the parallel interface must begin within the specified transmit latency.

The timing diagram below shows example timing for beginning loopback. In this example, the receiver is receiving a repeating stream of bytes, Rx-a thru Rx-z. Similarly, the MAC is causing the PHY to transmit a repeating stream of bytes Tx-a thru Tx-z. When the MAC asserts *TxDetectRx/Loopback* to the PHY, the PHY begins to loopback the received data to the differential *Tx+/Tx-* lines. Timing between assertion of *TxDetectRx/Loopback* and when Rx data is transmitted on the Tx pins is implementation dependent.



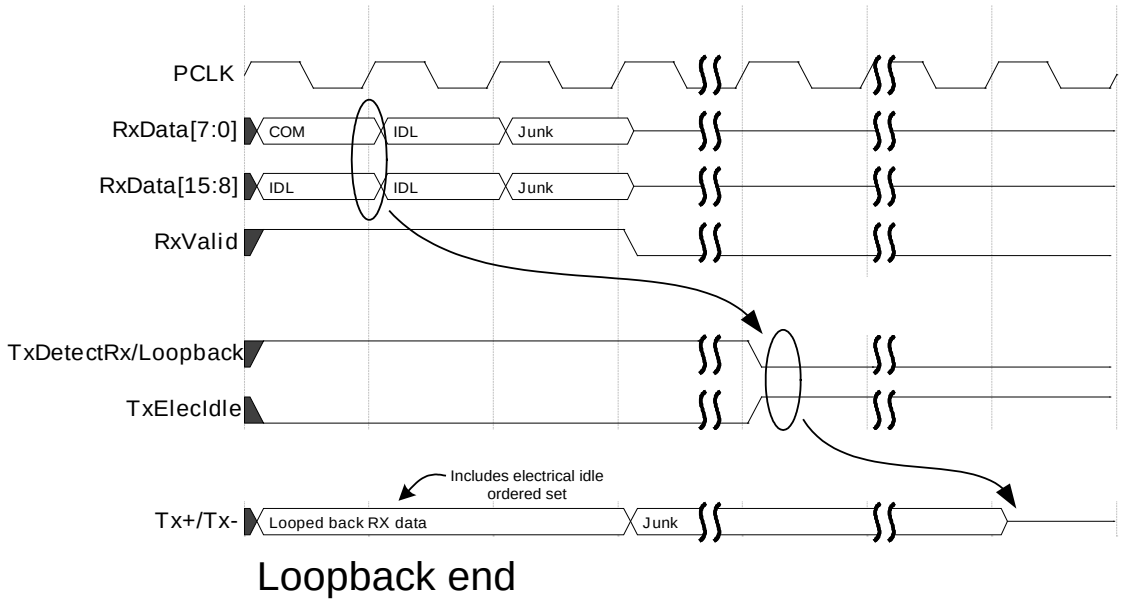
Loopback start

The next timing diagram shows an example of switching from loopback mode to normal mode when the PHY is operating in PCI Express Mode.

In PCI Express Mode, when the MAC detects an electrical idle ordered-set, the MAC deasserts *TxDetectRx/Loopback* and asserts *TxElecIdle*. The PHY must transmit at least three bytes of the

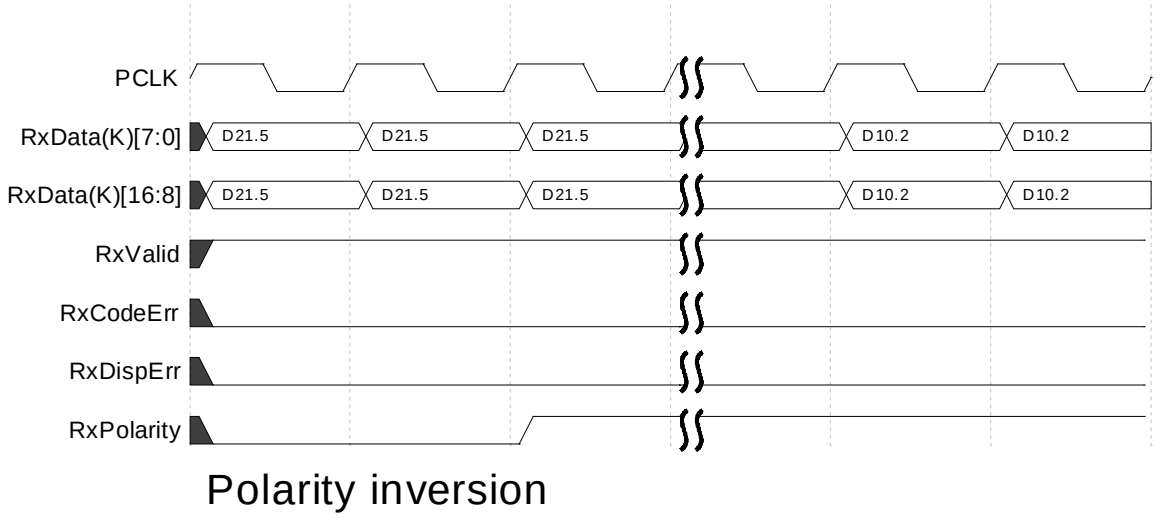
electrical idle ordered-set before going to electrical idle. (Note, transmission of the electrical idle ordered-set should be part of the normal pipeline through the PHY and should not require the PHY to detect the electrical idle ordered-set). The base spec requires that a Loopback Slave be able to detect and react to an electrical idle ordered set within 1ms. The PHY's contribution to this time consists of the PHY's Receive Latency plus the PHY's Transmit Latency (see section 6.13).

When the PHY is operating in USBMode, the device shall only transition out of loopback on detection of LFPS signaling (reset) or when VBUS is removed. When valid LFPS signaling is detected, the MAC transitions the PHY to the P2 power state in order to begin the LFPS handshake.



8.17 Polarity Inversion – PCI Express and USBModes

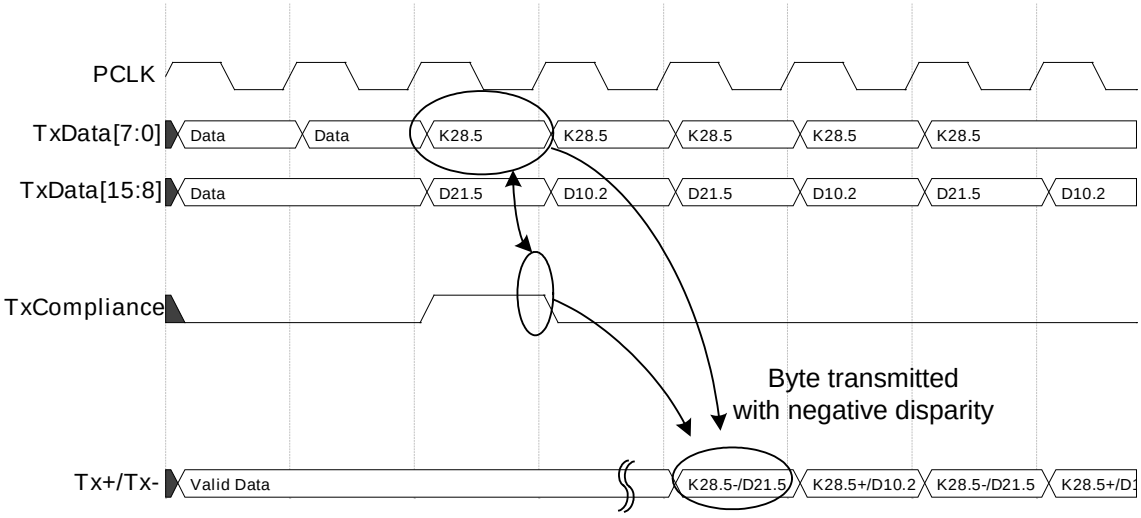
To support lane polarity inversion, the PHY must invert received data when *RxPolarity* is asserted. Inverted data must begin showing up on *RxData[]* within 20 PCLKs of when *RxPolarity* is asserted.



Polarity inversion

8.18 Setting negative disparity (PCI Express Mode)

To set the running disparity to negative, the MAC asserts *TxCompliance* for one clock cycle that matches with the data that is to be transmitted with negative disparity. For a 16-bit interface, the low order byte will be the byte transmitted where running disparity is negative. The example shows how *TxCompliance* is used to transmit the PCI Express compliance pattern in PCI Express mode. *TxCompliance* is only used in PCI Express mode and is qualified by *TxDataValid* when *TxDataValid* is being used.



Setting negative disparity

8.19 Electrical Idle – PCI Express Mode

The base spec requires that devices send an Electrical Idle ordered set before Tx+/Tx- goes to the electrical idle state. For a 16-bit interface or 32-bit interface, the MAC must always align the electrical idle ordered set on the parallel interface so that the COM symbol is on the low-order data lines (*TxDataK[7:0]).* Figure 8-6 shows an example of electrical idle exit and entry for a PCI Express 8 GT/s or 16 GT/s interface. TxDataValid must be asserted whenever TxElecIdle toggles as it is used as a qualifier for sampling TxElecIdle. Note: For SerDes architecture, 1 bit of TxElecIdle is required per 16-bits of data.

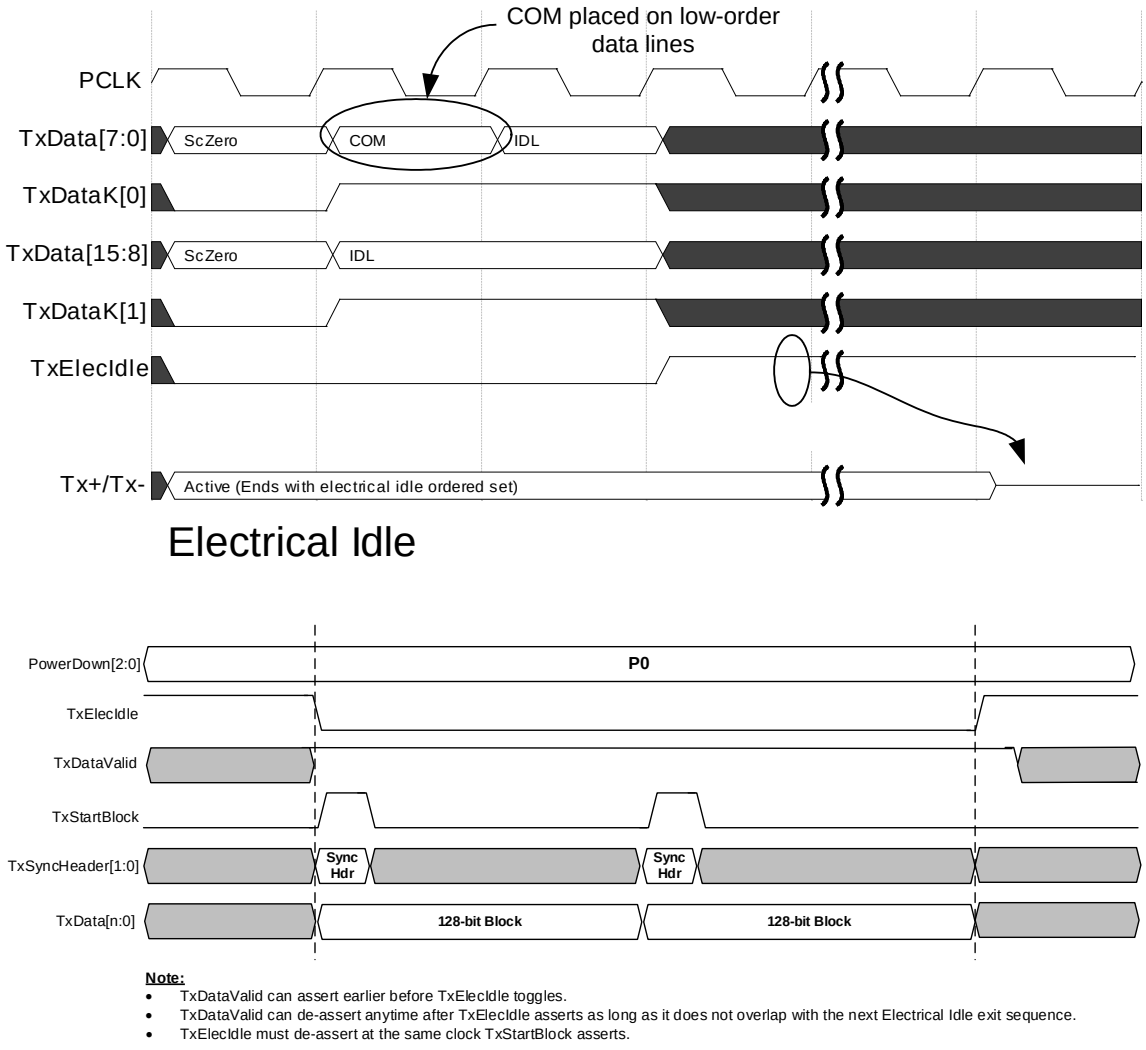


Figure 8-6 – PCI Express 3.0 TxDataValid Timings for Electrical Idle Exit and Entry.

Note: Figure 8-6 only shows two blocks of TxData and thus TxDataValid does not de—assert during the data. Other examples in the specification show longer sequences where TxDataValid de-asserts.

When data throttling is happening, TxElecIdle must be set long enough to be sampled by TxDataValid as shown in Figure 8-7.

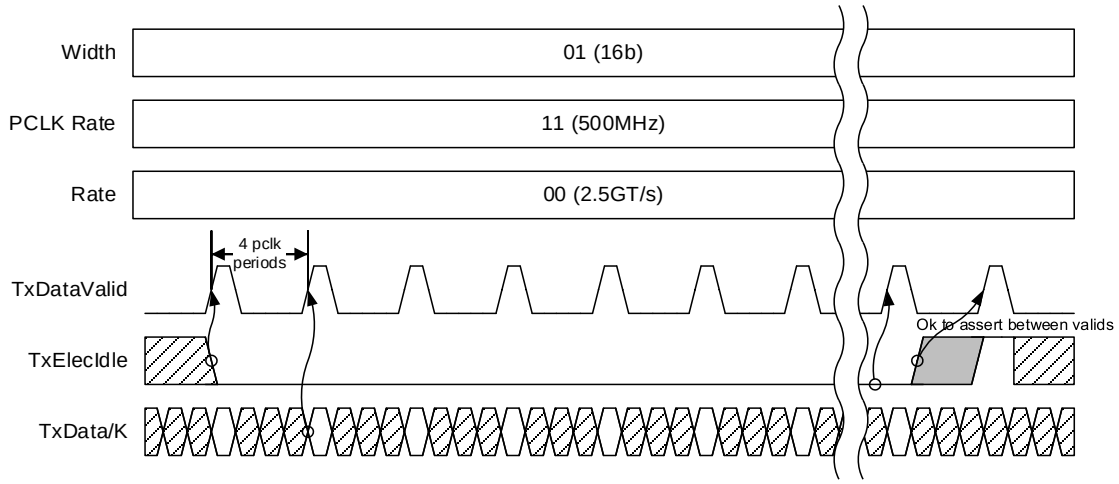


Figure 8-7. Data Throttling and TxElecIdle

8.20 Link Equalization Evaluation

While in the P0 power state, the PHY can be instructed to perform evaluation of the current TX equalization settings of the link partner. Basic operation of the equalization evaluation is that the MAC requests the PHY to evaluate the current equalization settings by asserting *RxEqEval*. When the PHY has completed evaluating the current equalization settings, it asserts *PhyStatus* for one clock and drives the *LinkEvaluationFeedback* signals to the appropriate feedback response. After link equalization evaluation has completed (as signaled by the assertion of *PhyStatus*), the MAC must deassert *RxEqEval* before initiating another evaluation. Figure 8-8 shows an example of the timings for a successful link equalization evaluation request. Figure 8-9 shows an example of the timings for a link equalization evaluation request resulting in feedback that is an invalid request.

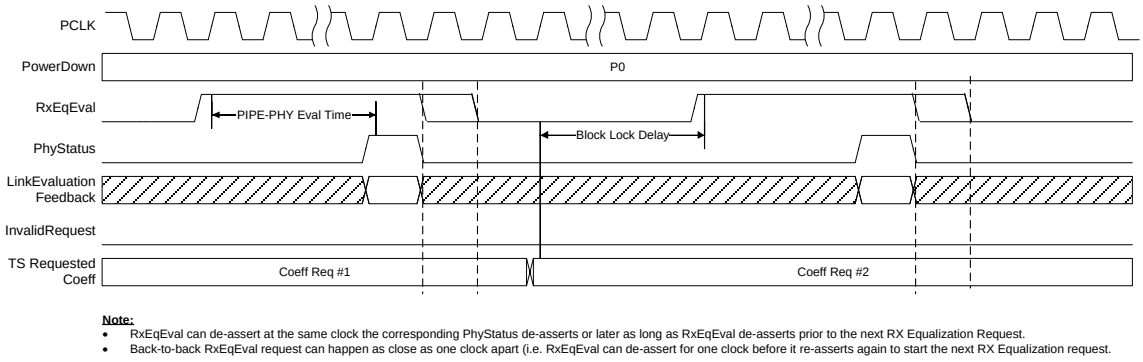


Figure 8-8 – PCI Express 8GT/s or higher Successful Equalization Evaluation Request

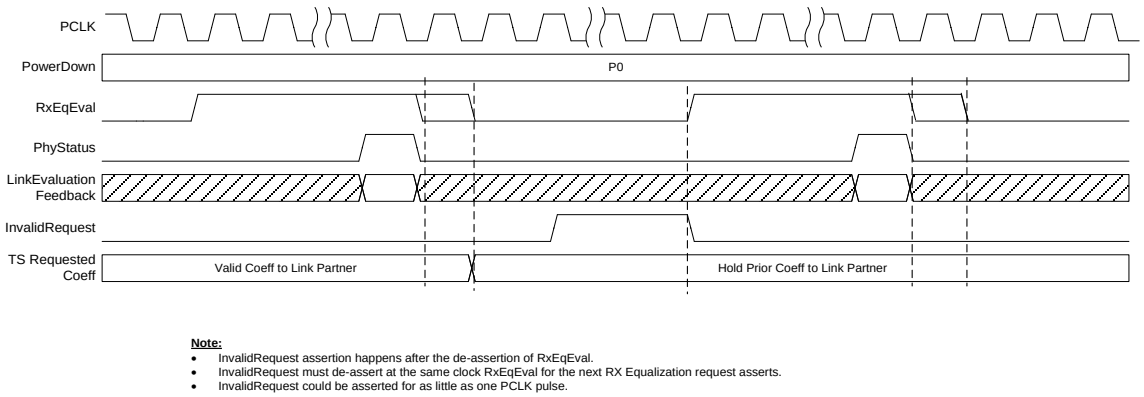


Figure 8-9 – PCI Express 3.0 Equalization Evaluation Request Resulting in Invalid Feedback

Once the MAC has requested link equalization evaluation (by asserting *RxEqEval*), the MAC must leave *RxEqEval* asserted until after the PHY has signaled completion by the assertion of *PhyStatus* unless the MAC needs to abort the evaluation due to high level timeouts or error conditions. To abort an evaluation the MAC de-asserts RxEqEval before the PHY has signaled completion. If the MAC aborts the evaluation the PHY must signal completion as quickly as possible. The MAC ignores returned evaluation values in an abort scenario.

Note: If a race condition occurs where the MAC aborts by deasserting RxEqEval on same cycle as the PHY asserts PhyStatus then the PHY shall not take any further action.

8.21 Implementation specific timing and selectable parameter support

PHY vendors (macrocell or discrete) must specify typical and worst case timings for the cases listed in Table 8-1. Other implementation specific parameters listed in Table 8-1 must also be specified advertised by the PHY in its datasheet.

Table 8-1 Parameters Advertised in PHY Datasheet

Transmit Latency	Time for data moving between the parallel interface and the PCI Express, SATA or USB serial lines. Timing is measured from when the data is transferred across the parallel interface (i.e. the rising edge of *PCLK*) and when the first bit of the equivalent 10-bit symbol is transmitted on the *Tx+/Tx-* serial lines. The PHY reports the latency for each operational mode the PHY supports. Note: If the transmit latency is different when EncodeDecodeBypass is asserted – the PHY must report this latency separately.
Receive Latency	Time for data moving between the parallel interface and the PCI Express, SATA or USB serial lines. Timing is measured from when the first bit of a 10-bit symbol is available on the

	Rx+/Rx- serial lines to when the corresponding 8-bit data is transferred across the parallel interface (i.e. the rising edge of *PCLK*). The PHY reports the latency for each operational mode the PHY supports. The reported latency is the nominal latency assuming the elasticity buffer is full to its nominal operating level. Note: If the receive latency is different when EncodeDecodeBypass is asserted – the PHY must report this latency separately. Additionally, the expected latency must be reported separately for both elasticity buffer operating modes.
Power State After Reset	The PHY power state immediately following reset. The state after reset needs to provide PCLK and have common mode off. Reporting this parameter is required if the PHY supports either SATA mode or PCI Express mode at 8 GT/s.
Loopback enable latency	Amount of time it takes the PHY to begin looping back receive data. Timed from when *TxDetectRx/Loopback* is asserted until the receive data is being transmitted on the serial pins. The PHY reports the latency for each operational mode the pHY supports.
Transmit Beacon – PCI Express Mode.	Timed from when the MAC directs the PHY to send a beacon (power state is P2 and *TxElecIdle* is deasserted) until the beacon signaling begins at the serial pins.
Receive Beacon – PCI Express Mode	Timed from when valid beacon signaling is present at the receiver pins until *RxElecIdle* is deasserted.
Transmit LFPS – USB Mode	Timed from when the MAC directs the PHY to send LFPS signaling until the LFPS signaling begins at the serial pins. Times are reported for each possible P state if the times are different for different power states.
Receive LFPS – USB Mode	Timed from when valid LFPS signaling is present at the receiver pins until *RxElecIdle* is deasserted.
N_FTS with common clock (PCI Express Mode)	Number of FTS ordered sets required by the receiver to obtain reliable bit and symbol lock when operating with a common clock. Note: This value may be required to be reported separately per rate.
N_FTS without common clock (PCI Express Mode)	Number of FTS ordered sets required by the receiver to obtain reliable bit and symbol lock when operating without a common

	clock. Note: This value may be required to be reported separately per rate.
PHY lock time	Amount of time for the PHY receiver to obtain reliable bit and symbol lock after valid symbols are present at the receiver. The PHY reports the time for each operational mode the PHY supports.
P0s to P0 transition time PCI Express Mode.	Amount of time for the PHY to return to P0 state, after having been in the P0s state. Time is measured from when the MAC sets the *PowerDown* signals to P0 until the PHY asserts *PhyStatus*. PHY asserts *PhyStatus* when it is ready to begin data transmission and reception.
P1 to P0 transition time. PCI Express Mode.	Amount of time for the PHY to return to P0 state, after having been in the P1 state. Time is measured from when the MAC sets the *PowerDown* signals to P0 until the PHY asserts *PhyStatus*. PHY asserts *PhyStatus* when it is ready to begin data transmission and reception.
P2 to P0 transition time PCI Express Mode.	Amount of time for the PHY to go to P0 state, after having been in the P2 state. Time is measured from when the MAC sets the *PowerDown* signals to P1 until the PHY deasserts *PhyStatus*.
P1 to P0 transition time. USB Mode.	Amount of time for the PHY to return to P0 state, after having been in the P1 state. Time is measured from when the MAC sets the *PowerDown* signals to P0 until the PHY asserts *PhyStatus*. PHY asserts *PhyStatus* when it is ready to begin data transmission and reception.
P2 to P0 transition time. USB Mode.	Amount of time for the PHY to return to P0 state, after having been in the P2 state. Time is measured from when the MAC sets the *PowerDown* signals to P0 until the PHY asserts *PhyStatus*. PHY asserts *PhyStatus* when it is ready to begin data transmission and reception.
P3 to P0 transition time USB Mode.	Amount of time for the PHY to go to P0 state, after having been in the P3 state. Time is measured from when the MAC sets the *PowerDown* signals to P0 until the PHY deasserts *PhyStatus*. PHY asserts *PhyStatus* when it is ready to begin data transmission and reception.
Power state transition times between two power states that provide PCLK.	Amount of time for the PHY to transition to a new power state. Time is measured from when the MAC sets the *PowerDown* signals to POWER_STATE_X until the PHY asserts *PhyStatus*. PHY asserts *PhyStatus* when it is ready to begin data transmission and reception. The PHY reports this transition between each

	pair of power states it supports in each PHY mode it supports.
Power state transition times between a power state without PCLK and a power state with PCLK.	Amount of time for the PHY to go to a power state providing PCLK, after having been in a power state that does not provide PCLK. Time is measured from when the MAC sets the *PowerDown* signals to the new power state until the PHY deasserts *PhyStatus*. The PHY reports this time for each possible transition between a power state that does not provide PCLK and a power state that does provide PCLK. The PHY reports this transition time between each pair of power states it supports in each PHY mode it supports.
Power state transition times between a power state without PCLK and a power state without PCLK.	Amount of time for the PHY to go to a power state without PCLK, after having been in a power state that does not provide PCLK. Time is measured from when the MAC sets the *PowerDown* signals to the new power state until the PHY deasserts *PhyStatus*. The PHY reports this time for each possible transition between a power state that does not provide PCLK and a power state that does not provide PCLK. The PHY reports this transition time between each pair of power states it supports in each PHY mode it supports.
Supported power states.	The PHY lists each power state it supports for each PHY mode it supports. For each power state supported it reports whether PCLK is provided, the exit latency to the active power state, whether RxElecIdle is supported in the state, and the common mode state. Note: This is done for all states not already listed separately.
L1 Substate Management Mechanism	The PHY reports which of the following mechanisms it supports for L1 substate management: 1) Exclusively managed via PowerDown[3:0] 2) Managed via RxEIDetectDisable and TxCommonModeDisable 3) Both of the above mechanisms are supported
Simultaneous Rate and Power State Change	The PHY reports if it supports simultaneous rate and power state changes for each PHY mode it supports.
Data Rate change time. PCI Express Mode and SATA Mode.	Amount of time the PHY takes to perform a data rate change. Time is measured from when the MAC changes *Rate* to when the PHY signals rate change complete with the single clock assertion

<table>
<tr><td></td><td>of PhyStatus. There may be separate values for each possible change between different supported rates for each supported PHY mode.</td></tr>
<tr><td>Transmit Margin values supported. PCI Express Mode and USB Mode.</td><td>Transmitter voltage levels.
<table>
<tr><th>[2]</th><th>[1]</th><th>[0]</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>TxMargin value 0 =</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>TxMargin value 1 =</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>TxMargin value 2 =</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>TxMargin value 3 =</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>TxMargin value 4 =</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>TxMargin value 5 =</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>TxMargin value 6 =</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>TxMargin value 7 =</td></tr>
</table></td></tr>
<tr><td>Max Equalization Settings for C_{-1}</td><td>Reports the maximum number of settings supported by the PHY for the 8.0 GT/s and 16 GT/s equalization. The maximum number of settings must be less than 64.</td></tr>
<tr><td>Max Equalization Settings for C_0</td><td>Reports the maximum number of settings supported by the PHY for the 8.0 GT/s and 16 GT/s equalization. The maximum number of settings must be less than 64.</td></tr>
<tr><td>Max Equalization Settings for C_1</td><td>Reports the maximum number of settings supported by the PHY for the 8.0 GT/s and 16 GT/s equalization. The maximum number of settings must be less than 64.</td></tr>
<tr><td>Default Equalization settings for full swing preset Pn.</td><td>Reports the recommended setting values for C_{-1}, C_0, C_1 for each full swing preset. Note: This should be reported separately per rate.</td></tr>
<tr><td>Default Equalization settings for half swing preset Pn.</td><td>Reports the recommended setting values for C_{-1}, C_0, C_1 for each half swing preset. Note: This should be reported separately per rate.</td></tr>
<tr><td>Default Equalization settings for recommended TX EQ value of 0 dB preshoot and -2.5 dB de-mephasis.</td><td>Reports the recommended setting values for C_{-1}, C_0, C_1 for the USB 3.1 0 dB preshoot and -2.5 dB de-emphasis recommended TX EQ setting.</td></tr>
<tr><td>Default Equalization settings for recommended TX EQ value of 2.7 dB preshoot and -3.3 dB de-mephasis.</td><td>Reports the recommended setting values for C_{-1}, C_0, C_1 for the USB 3.1 0 dB preshoot and -2.5 dB de-emphasis recommended TX EQ setting.</td></tr>
<tr><td>Dynamic Preset Coefficient Update Support</td><td>A PHY indicates if it dynamically updates coefficients.</td></tr>
<tr><td>Figure of Merit range</td><td>If the PHY reports link equalization feedback in the Figure of Merit format it reports the maximum value it will report. The maximum value must be less than 256.</td></tr>
<tr><td>Figure of Merit for BER target</td><td>If the PHY reports link equalization feedback in the Figure of Merit format it reports the minimum value that the PHY estimates corresponds to a link BER of E-12.</td></tr>
<tr><td>Default Link Partner Preset[3:0]</td><td>If the PHY prefers the link parter to start with a specific preset during link evaluation it reports the preferred starting preset.</td></tr>
</table>

	The default link partner preset value is encoded as follows: 0000b – Preset P0. 0001b – Preset P1. 0010b – Preset P2. 0011b – Preset P3. 0100b – Preset P4. 0101b – Preset P5. 0110b – Preset P6. 0111b – Preset P7. 1000b – Preset P8. 1001b – Preset P9. 1010b – Preset P10. 1011b – Reserved 1100b – Reserved 1101b – Reserved 1110b – Reserved 1111b – No Preference. Note: This should be reported separately per rate.
Beacon Support	The PHY indicates whether it supports beacon transmission. Beacon transmission is optional. 1: Beacon transmission is supported. 0: Beacon transmission is not supported.
EncodeDecodeBypassSupport[3:0]	The PHY indicates whether it supports optional EncodeDecodeBypass mode at each signaling rate. [0] Rate[1:0] = 0 [1] Rate[1:0] = 1 [2] Rate[1:0] = 2 [3] Rate[1:0] = 3 The support value for each rate is encoded as follows: 0 - No support for EncodeDecodeBypass 1 – Support for EncodeDecodeBypass
NoDeemphasisSupport[1:0]	The PHY indicates whether it supports an optional No De-emphasis signaling mode at 2.5 and 5.0 GT/s signaling rates. [0] Support at 2.5 GT/s [1] Support at 5.0 GT/s The support value for each rate is encoded as follows: 0 – No support for a no de-emphasis signaling mode.

	1 – Support for a no de-emphasis signaling mode.
SupportedLFPresets	List of presets the PHY supports at 8 GT/s and 16 GT/s for half swing in addition to the 5 required by the base spec.
PCLK Mode[1:0]	The PHY indicates whether it support PCLK as a PHY output or PCLK as a PHY input. [0] Supports PCLK as an output [1] Supports PCLK as an input The support value for each rate is encoded as follows: 0 – No support. 1 – Support. Configuration for a PHY that supports both PCLK modes is PHY specific.
PHYClockInsertionDelay	A PHY that supports “PCLK as an input” mode must report the maximum delay and the minimum delay (insertion delay) for any sequential logic at the MAC/PHY interface that will use PCLK in the PHY in picoseconds.
SupportedPhyModes	List of all modes the PHY supports for the PHY Mode[1:0] input.
MaximumPCIExpressRate	Value for DataRate input corresponding to the maximum rate the PHY supports while in PCI Express mode. This field is undefined if the PHY does not support PCI Express mode.
MaximumSataRate	Value for the DataRate input corresponding to the maximum rate the PHY supports while in Sata Mode. This field is undefined if the PHY does not support Sata mode.
ListofSupportedSataModes	List of all supported signaling rate, width, PCLK rate combinations supported in Table 3-2.
ListofSupportedPCIExpressModes	List of all supported signaling rate, width, PCLK rate combinations supported in Table 3-1.
MaximumEntriesInElasticityBuffer	Maximum number of entries that can be stored in the elasticity buffer. The PHY reports the maximum number of entries for each operational mode the PHY supports.
ElasticityBufferEntrySize	Size of a data entry in the elasticity buffer in bits. The PHY reports this size for each operation mode the PHY supports.
EnhancedPTMTimingSupport	The PHY indicates whether it supports optional elasticity buffer location information through the ElasticBufferLocation control signals to allow more accurate timing of received packets within

	the MAC. The support value is encoded as follows: 0 – No support. 1 – Support.
L1PMSubStatesSupport	The PHY indicates whether it supports optional L1 PM Substates. A PHY which supports L1 PM Substates must support asynchronous power state transitions. The support value is encoded as follows: 0 – No support. 1 – Support.
RXMarginingVoltageSupported[2]	The PHY indicates whether it supports voltage margining, encoded as follows: 0 – No Support 1 – Support.
RXMarginingSamplingRateVoltage[5:0][2]	Percentage of bits margined during voltage margining mode is calculated as 1/64*(Sampling_Rate[5:0]+1). Allowable values: 0-63.
RXMarginingSamplingRateTiming[5:0][2]	Percentage of bits margined during timing margining mode is calculated as 1/64*(Sampling_Rate[5:0]+1). Allowable values: 0-63.
RXMarginingIndependentLeftRight[2]	The PHY indicates whether it supports independent left and right time margining. The support value is encoded as follows: 0 – No Support 1 – Support.
RXMarginingIndependentUpDown[2]	The PHY indicates whether it supports independent up and down voltage margining. The support value is encoded as follows: 0 – No Support 1 – Support.
RXMarginingIndependentErrorSampler[2]	The PHY indicates whether it supports an error sampler independent from the main sampler to allow higher BER's to be measured. The support value is encoded as follows: 0 – No Support 1 – Support.
RXMarginingVoltageSteps[6:0][2]	Total number of voltage steps, minimum range +/- 50mV. A value of zero indicates that voltage margining is not supported. Allowable non-zero values: 32-127.
RXMarginingTimingSteps[5:0][2]	Total number of timing steps, minimum range +/-0.2UI. Allowable values: 8-63.

[2] See PCIe Base Specification. In case of discrepancy, the PCIe Base Specification shall supercede the PIPE specification.

RXMarginingMaxVoltageOffset[6:0][2]	Offset at maximum step value as percentage of one volt. Allowable values: 5-50.
RXMarginingMaxTimingOffset[6:0][2]	Offset at maximum step value as percentage of nominal UI. Allowable values: 20-50.
RXMarginingMaxLanes[5:0][2]	Maximum number of lanes that can be margined simultaneously. Allowable values:1-32. Recommended value=number of lanes the PHY supports.
RXMarginingSampleReportingMethod[2]	Indicates whether a sample frequency or a sample count is reported. This value is encoded as follows: 0 – Sample Count Reported 1 – Sample Frequency Reported
RXMarginingMaxTimingOffsetChange[6:0]	Maximum number of steps margin offset can be changed with one command during timing margining. Allowable values: 1-127.
RXMarginingMaxVoltageOffsetChange[6:0]	Maximum number of steps margin offset can be changed with one command during voltage margining. Allowable values: 1-127.
RXMessageBusWriteBufferDepth[3:0]	The PHY indicates the number of write buffer entries that it has implemented to receive writes from the MAC, where one entry can hold the three bytes of information associated with each write transaction.
TXMessageBusMinWriteBufferDepth[3:0]	The PHY indicates the minimum number of write buffer entries it expects the MAC to implement to receive writes from the PHY. Allowable values: 0-8. The MAC may choose to implement more than the minimum required by the PHY; however, there may not be any benefit in doing so.

8.22 Control Signal Decode table – PCI Express Mode

The following table summarizes the encodings of four of the seven control signals that cause different behaviors depending on power state. For the other three signals, *Reset#* always overrides any other PHY activity. *TxCompliance* and *RxPolarity* are only valid when the PHY is in P0 and is actively transmitting. Note that these rules only apply to lanes that have not been 'turned off' as described in section 8 (Multi-lane PIPE).

PowerDown[1:0]	*TxDetectRx/ Loopback*	*TxElecIdle*	Description
P0: 00b	0	0	PHY is transmitting data. MAC is providing data bytes to be sent every clock cycle.
	0	1	PHY is not transmitting and is in electrical idle.
	1	0	PHY goes into loopback mode.
	1	1	Illegal. MAC should never do this.

P0s: 01b	Don't care	0	Illegal. MAC should always have PHY doing electrical idle while in P0s. PHY behavior is undefined if *TxElecIdle* is deasserted while in P0s or P1.
		1	PHY is not transmitting and is in electrical idle. Note that any data transferred across the PIPE interface before *TxElecIdle* is asserted, but not yet signaled on the analog interface is signaled before the analog interface becomes idle.
P1: 10b	Don't care	0	Illegal. MAC should always have PHY doing electrical idle while in P1. PHY behavior is undefined if *TxElecIdle* is deasserted while in P0s or P1.
	0	1	PHY is idle.
	1	1	PHY does a receiver detection operation.
P2: 11b	Don't care	0	PHY transmits Beacon signaling
		1	PHY is idle.

8.23 Control Signal Decode table – USB Mode and Converged IO Mode

The following table summarizes the encodings of four of the seven control signals that cause different behaviors depending on power state. For the other three signals, *Reset#* always overrides any other PHY activity. *RxPolarity* is only valid, and therefore should only be asserted, when the PHY is in P0 and is actively transmitting.

PowerDown[1:0]	*TxDetectRx/ Loopback*	*TxElecIdle*	Description
P0: 00b	0	0	PHY is transmitting data. MAC is providing data bytes to be sent every clock cycle.
	0	1	PHY is not transmitting and is in electrical idle. Note that any data transferred across the PIPE interface before *TxElecIdle* is asserted, but not yet signaled on the analog interface is signaled before the analog interface becomes idle.
	1	0	PHY goes into loopback mode.
	1	1	PHY transmits LFPS signaling.
P1: 01b	Don't care	0	PHY transmits LFPS signaling
		1	PHY is not transmitting and is in electrical idle.
P2: 10b or P3: 11b	Don't care	0	PHY transmits LFPS signaling
	0	1	PHY is idle.
	1	1	PHY does a receiver detection operation.

8.24 Control Signal Decode table – SATA Mode

The following table summarizes the encodings of the control signals that cause different behaviors in POWER_STATE_0. For other control signals, *Reset#* always overrides any other PHY activity.

Note: The PHY transmit latency reported in section 8.210must be consistent for all the different behaviors in POWER_STATE_0. This means that the amount of time OOB signaling is present on the analog TX pair must be the same as the time OOB signaling was indicated on the PIPE interface.

PowerDown[2:0]	*TxDetectRx/ Loopback*	*TxElecIdle*	Description
POWER_STATE_0: 00b	0	0	PHY is transmitting data. MAC is providing data bytes to be sent every clock cycle.
	0	1	PHY is not transmitting and is in electrical idle. Note that any data transferred across the PIPE interface before *TxElecIdle* is asserted, but not yet signaled on the analog interface is signaled before the analog interface becomes idle.
	1	0	PHY goes into loopback mode.

	1	1	PHY transmits OOB signaling with pattern determined by TX Pattern. Note that a PHY must ensure the transition between OOB signaling and data signaling is performed smoothly on a symbol boundary on the analog interface.
Power Stater other than POWER_STATE_0	Don't care	Don't care	PHY is not transmitting and is in electrical idle.
			PHY is not transmitting and is in electrical idle.

8.25 Required synchronous signal timings

To improve interoperability between MACs and PHYs from different vendors the following timings for synchronous signals are required:

Setup time for input signals	No greater than 25% of cycle time
Hold time for input signals	0ns
PCLK to data valid for outputs	No greater than 25% of cycle time

8.26 128b/130b Encoding and Block Synchronization (PCI Express 8 GT/s and 16 GT/s)

For every block (usually 128 bits – shorter/longer SKP blocks are sometimes transmitted by Retimers) that is moved across the PIPE TxData interface at the 8.0 GT/s rate or 16 GT/s rate the PHY must transmit 2 extra bits. The MAC must use the TxDataValid signal periodically to allow the PHY to transmit the built up backlog of data. For example – if the TxData bus is 16 bits wide and PCLK is 500 Mhz then every 8 blocks the MAC must deassert TxDataValid for one PCLK to allow the PHY to transmit the 16 bit backlog of built up data. The buffers used by the PHY to store TX data related to the 128/130b encoding rate mismatch must be empty when the PHY comes out of reset and must be empty whenever the PHY exits electrical idle (since TX buffers are flushed before entry to idle). The PHY must use RxDataValid in a similar fashion. TxDataValid and RxDataValid must be de-asserted for one clock exactly every N blocks when the PIPE interface is operating at 8 GT/s or 16 GT/s, where N is 4 for an 8 bit wide interface, 8 for a 16 bit wide interface, and 16 for a 32 bit wide interface. The MAC must first de-assert TxDataValid immediately after the end of the Nth transmitted block following reset or exit from electrical idle. Examples of the timing for TxDataValid are shown in Figure 8-10 for a 8 bit interface and in Figure 8-11 for a 16 bit interface. The PHY must first de-assert RxDataValid immediately after the end of the Nth received block transmitted across the PIPE interface following reset or exit from electrical idle. Examples of timings for RxDataValid and other Rx related signals for a 16 bit wide interface are shown in Figure 8-12.

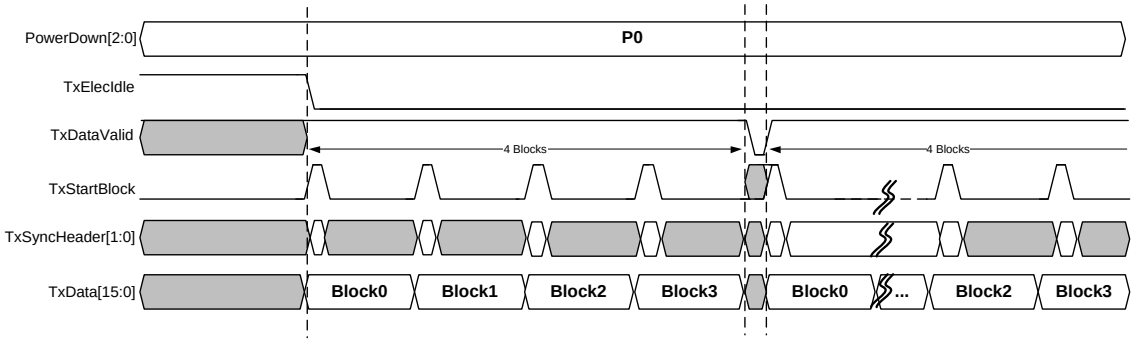


Figure 8-10 – PCI Express 8 GT/s or higher TxDataValid Timing for 8 Bit Wide TxData Interface

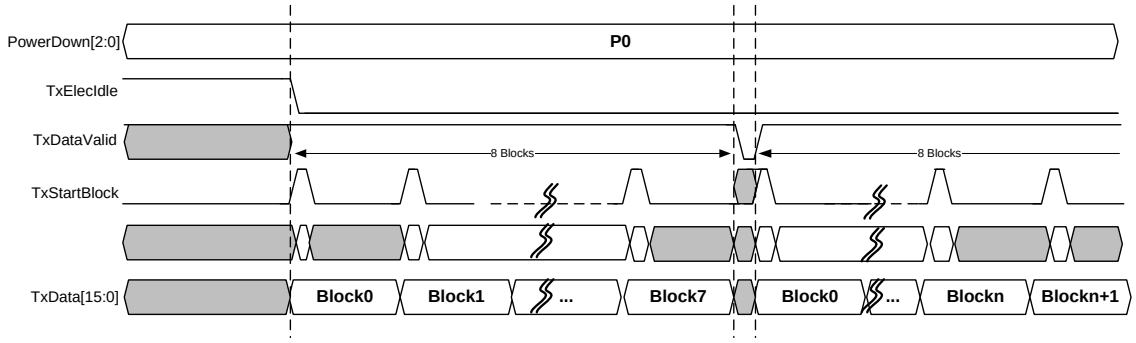
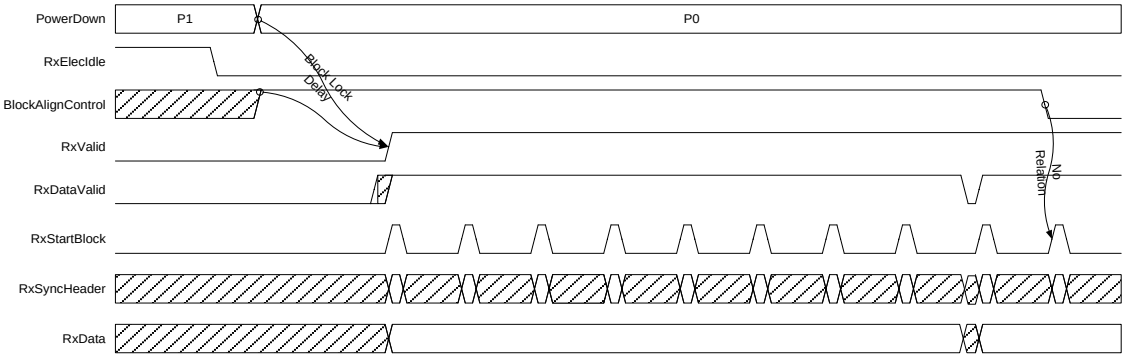


Figure: TxDataValid Timing for 16-bit TxData Interface

Figure 8-11 – PCI Express 8 GT/s or higher TxDataValid Timing for 16 Bit Wide TxData Interface



Notes:

- RxValid assertion indicates that PHY has achieved block alignment.
- RxValid assertion aligns with the first RxStartBlock.
- RxDataValid can assert before RxValid toggles or at the latest the same clock when RxValid toggles.
- There is no required relationship between BlockAlignControl de-assertion and RxStartBlock.

Figure 8-12 – PCI Express 8 GT/s or higher RxDataValid Timing for 16 Bit Wide RxData Interface

There are situations, such as upconfigure, when a MAC must start transmissions on idle lanes while some other lanes are already active. In any such situation the MAC must wait until the

cycle after TxDataValid is de-asserted to allow the PHY to transmit the backlog of data due to 128b/130b to start transmissions on previously idle lanes.

8.27 128b/132b Encoding and Block Synchronization (USB 10 GT/s)

For every 128 bits that are moved across the PIPE TxData interface at the 10.0 GT/s rate the PHY must transmit 132 bits. The MAC must use the TxDataValid signal periodically to allow the PHY to transmit the built up backlog of data. For example – if the TxData bus is 16 bits wide and PCLK is 625 Mhz then every 4 blocks the MAC must deassert TxDataValid for one PCLK to allow the PHY to transmit the 16 bit backlog of built up data. The buffers used by the PHY to store TX data related to the 128/132b encoding rate mismatch must be empty when the PHY comes out of reset and must be empty whenever the PHY exits electrical idle (since TX buffers are flushed before entry to idle). The PHY must use RxDataValid in a similar fashion. TxDataValid and RxDataValid must be de-asserted for one clock exactly every N blocks when the PIPE interface is operating at 10 GT/s, where N is 2 for an 8 bit wide interface, 4 for a 16 bit wide interface, and 8 for a 32 bit wide interface. The MAC must first de-assert TxDataValid immediately after the end of the Nth transmitted block following reset or exit from electrical idle.

8.28 Message Bus Interface

8.28.1 General Operational Rules

The message bus interface can be used after Reset# is deasserted and PCLK is stable. The message bus interface must return to its idle state immediately upon assertion of Reset# and must remain idle until Reset# is deasserted and PhyStatus is deasserted . Since the MAC is aware of when PCLK is stable, the requirement that PCLK must be an input to use the message bus allows the MAC to only issue transactions on the message bus after PCLK becomes stable. The expectation is that the MAC is always the first to initiate a transaction on the message bus after Reset# deassertion, and that the PHY only sends transactions in response, as part of a sequence initiated by the MAC.

For each write_committed issued, the initiator must wait for a write_ack response before issuing any new write_uncommitted or write_committed transactions. A sequence of write_uncommitted transactions must always be followed by a write_committed transaction; only a single write_ack response is expected. The initiator must ensure that the total number of outstanding writes, i.e. writes issued since the last write_ack was received, must not exceed the write buffer storage implemented by the receiver.

Only one read can be outstanding at a time in each direction. The initiator must wait for a read completion before issuing a new read since there are no transaction IDs associated with outstanding reads.

To facilitate design simplicity, reads and writes cannot be mixed. There must not be any reads outstanding when a write is issued; conversely, there must not be any writes outstanding when a read is issued. An outstanding write is any write_committed that hasn't received a write_ack or any write_uncommitted without a subsequent write_committed that has received a write_ack.

8.28.2 Message Bus Operations vs Dedicated Signals

For simplicity, dependencies between message bus operations and dedicated signals are kept to a minimum. The dependencies that do exist are there only because no acceptable workarounds for eliminating them have been identified; these dependencies are documented in this section:

- The PHY must wait for the write_ack to come back for any write to LocalLF, LocalFS, LocalG4LF, or LocalG4FS, if any, before it asserts PhyStatus for a rate change.

8.29 PCI Express Lane Margining at the Receiver

Table 8-2 provides the sequence of PIPE message bus commands associated with various receiver margining operations; different sequences are shown for independent and dependent samplers.

Table 8-2. Lane Margining at the Receiver Sequences

Operation	Type of Sampler	Sequence		
		Direction	**Msg Bus Cmd**	**Description**
Start Margining Success	independent	M-->P	UWr	RxMarginControl1={1'b?,7'b?} (direction, offset)
		M-->P	CWr	RxMarginControl0=8'b000011?1 (clear error/sample and set start)
				Mac clears its error count snapshot
		P-->M	Ack	
				PHY clears its error and sample counters due to MAC setting Sample Count Reset and Error Count Reset bits in RxMarginControl0
		P-->M	UWr	RxMarginStatus1.SampleCount=0
		P-->M	UWr	RxMarginStatus2.ErrorCount=0
		P-->M	CWr	RxMarginStatus0.MarginStatus=1
		M-->P	Ack	
	dependent	M-->P	UWr	RxMarginControl1={1'b?,7'b?} (direction, offset)
		M-->P	CWr	RxMarginControl0=8'b000011?1 (set start) (error/sample clears are a don't care)
				Mac clears its error count snapshot
		P-->M	Ack	
		P-->M	CWr	RxMarginStatus0.MarginStatus=1
		M-->P	Ack	
Offset Change Success	independent	M-->P	UWr	RxMarginControl0=8'b000011?1 (clear error/sample counts)
		M-->P	CWr	RxMarginControl1={1'b?,7'b?} (direction, offset)
				Mac clears its error count snapshot
		P-->M	Ack	
				PHY clears its error and sample counters due to MAC setting Sample Count Reset and Error Count Reset bits in RxMarginControl0
		P-->M	UWr	RxMarginStatus1.SampleCount=0
		P-->M	UWr	RxMarginStatus2.ErrorCount=0
		P-->M	CWr	RxMarginStatus0.MarginStatus=1
		M-->P	Ack	
	dependent	M-->P	CWr	RxMarginControl1={1'b?,7'b?} (direction, offset)

				Mac clears its error count snapshot
		P-->M	Ack	
		P-->M	CWr	RxMarginStatus0.MarginStatus=1
		M-->P	Ack	
Clear Error	independent	M-->P	CWr	RxMarginControl0=8'b000001?1 (clear error, hold t vs v, maintain start)
		P-->M	Ack	
		P-->M	UWr	RxMarginStatus1.SampleCount=current
		P-->M	CWr	RxMarginStatus2.ErrorCount=0
		M-->P	Ack	
	dependent			Mac clears its error count snapshot
Stop Margining	independent	M-->P	UWr	RxMarginControl1={1'b?,7'b?} (direction, offset)
		M-->P	CWr	RxMarginControl0=8'b00000000 (stop, clear t vs v)
		P-->M	Ack	
		P-->M	UWr	RxMarginStatus1.SampleCount=Final
		P-->M	UWr	RxMarginStatus2.ErrorCount=Final
		P-->M	CWr	RxMarginStatus0.MarginStatus=1
		M-->P	Ack	
	dependent	M-->P	UWr	RxMarginControl1={1'b?,7'b?} (direction, offset)
		M-->P	CWr	RxMarginControl0=8'b00000000 (stop, clear t vs v)
		P-->M	Ack	
		P-->M	CWr	RxMarginStatus0.MarginStatus=1
		M-->P	Ack	
Start Margining NAK	independent	M-->P	UWr	RxMarginControl1={1'b?,7'b?} (direction, offset)
		M-->P	CWr	RxMarginControl0=8'b000011?1 (clear error/sample and start)
				Mac clears its error count snapshot
		P-->M	Ack	
				PHY clears its error and sample counters due to MAC setting Sample Count Reset and Error Count Reset bits in RxMarginControl0
		P-->M	UWr	RxMarginStatus1.SampleCount=0
		P-->M	UWr	RxMarginStatus2.ErrorCount=0
		P-->M	CWr	RxMarginStatus0.MarginNak=1
		M-->P	Ack	
				MAC changes execution status to 11 (NAK)
	dependent	M-->P	UWr	RxMarginControl1={1'b?,7'b?} (direction, offset)
		M-->P	CWr	RxMarginControl0=8'b000011?1 (set start) (error/sample clears are a don't care)
				Mac clears its error count snapshot
		P-->M	Ack	
				PHY detects bad margin request, places/keeps margin logic in normal

				functional operation mode
		P-->M	CWr	RxMarginStatus0.MarginNak=1
		M-->P	Ack	
				MAC changes execution status to 11 (NAK)
Offset Change NAK	independent	M-->P	UWr	RxMarginControl0=8'b000011?1 (clear error/sample counts)
		M-->P	CWr	RxMarginControl1={1'b?,7'b?} (direction, offset)
				Mac clears its error count snapshot
		P-->M	Ack	
				"PHY clears its error and sample counters due to MAC setting Sample Count Reset and Error Count Reset bits in RxMarginControl0. PHY detects bad offset, places/keeps margin logic in normal functional operation mode (margin off)"
		P-->M	UWr	RxMarginStatus1.SampleCount=0
		P-->M	UWr	RxMarginStatus2.ErrorCount=0
		P-->M	CWr	RxMarginStatus0.MarginNak=1
		M-->P	Ack	
				MAC changes execution status to 11 (NAK)
	dependent	M-->P	CWr	RxMarginControl1={1'b?,7'b?} (direction, offset)
				Mac clears its error count snapshot
		P-->M	Ack	
				PHY detects bad offset, places/keeps margin logic in normal functional operation mode (margin off)
		P-->M	CWr	RxMarginStatus0.MarginNak=1
		M-->P	Ack	
				MAC changes execution status to 11 (NAK)
Error & Sample Counts Update (under limit)	independent			PHY detects a change in error or sample count (note: multiple updates may be combined into single write to avoid backlog)
		P-->M	UWr	RxMarginStatus1.SampleCount= new current
		P-->M	CWr	RxMarginStatus2.ErrorCount= new current
		M-->P	Ack	
				MAC changes execution status to new error/sample count
	dependent			MAC detects a change in error count
				MAC changes execution status to new error count
Error Limit Reached	independent			PHY detects a change in error or sample count
		P-->M	UWr	RxMarginStatus1.SampleCount= current
		P-->M	CWr	RxMarginStatus2.ErrorCount= new current
		M-->P	Ack	
				MAC compares error update to limit, detects limit reached

		M-->P	UWr	RxMarginControl1={1'b?,7'b?} (direction, offset)
		M-->P	CWr	RxMarginControl0=8'b00000000 (stop, clear t vs v)
		P-->M	Ack	
		P-->M	UWr	RxMarginStatus1.SampleCount=Final
		P-->M	UWr	RxMarginStatus2.ErrorCount=Final
		P-->M	CWr	RxMarginStatus0.MarginStatus=1
		M-->P	Ack	
				MAC changes execution status to 00 (error limit reached)
	dependent			MAC observes error count has reached limit
		M-->P	UWr	RxMarginControl1={1'b?,7'b?} (direction, offset)
		M-->P	CWr	RxMarginControl0=8'b00000000 (stop, clear t vs v)
		P-->M	Ack	
		P-->M	CWr	RxMarginStatus0.MarginStatus=1
		M-->P	Ack	
				MAC changes execution status to 00 (error limit reached)
Sample Count Saturated	independent			PHY detects a change in error or sample count
		P-->M	UWr	RxMarginStatus1.SampleCount= ==7'h7F
		P-->M	CWr	RxMarginStatus2.ErrorCount= new current
		M-->P	Ack	
				MAC sees sample count is 7'h7F (all 1s)
		M-->P	UWr	RxMarginControl1={1'b?,7'b?} (direction, offset)
		M-->P	CWr	RxMarginControl0=8'b00000000 (stop, clear t vs v)
		P-->M	Ack	
		P-->M	UWr	RxMarginStatus1.SampleCount=Final
		P-->M	UWr	RxMarginStatus2.ErrorCount=Final
		P-->M	CWr	RxMarginStatus0.MarginStatus=1
		M-->P	Ack	
	dependent			N/A

9 Sample Operational Sequences

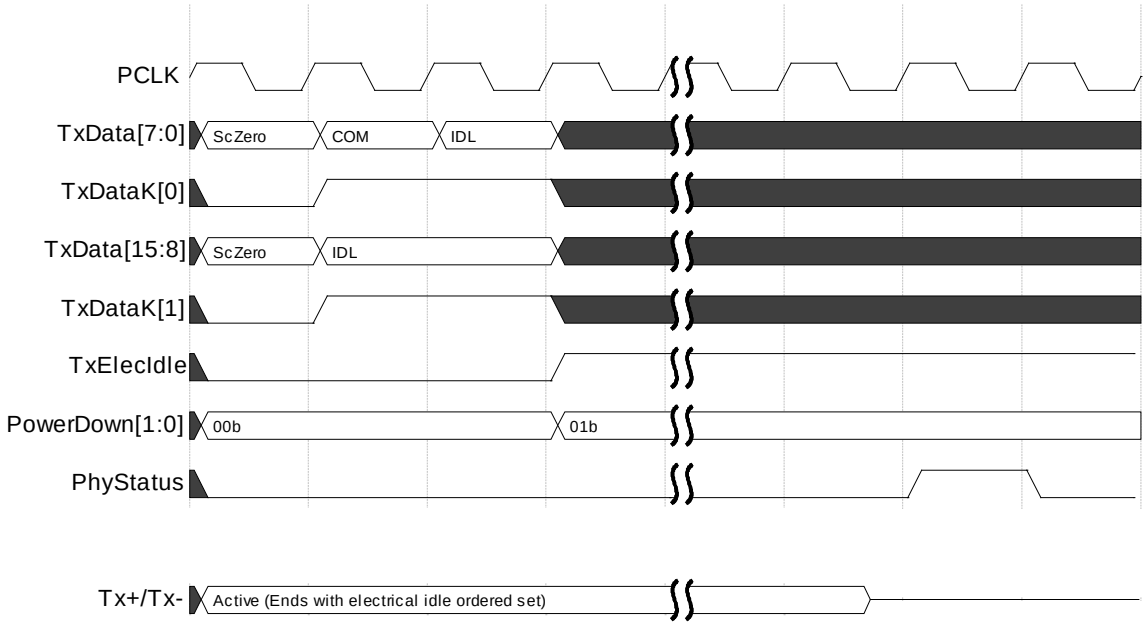
These sections show sample timing sequences for some of the more common PCI Express, SATA and USB operations. These are *sample* sequences and timings and are not required operation.

9.1 Active PM L0 to L0s and back to L0 – PCI Express Mode

This example shows one way a PIPE PHY can be controlled to perform Active State Power Management on a link for the sequence of the link being in L0 state, transitioning to L0s state, and then transitioning back to L0 state.

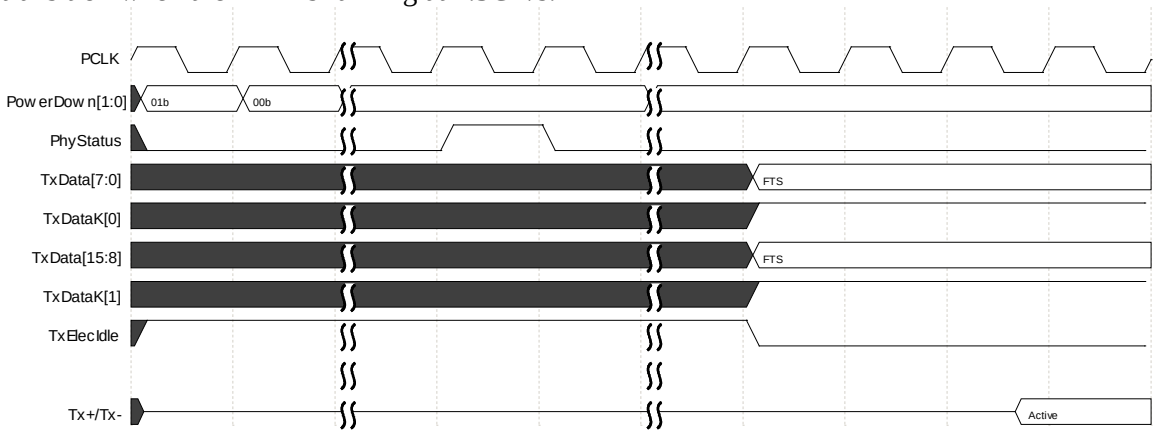
When the MAC and higher levels have determined that the link should transition to L0s, the

MAC transmits an electrical idle ordered set and then has the PHY transmitter go idle and enter P0s. Note that for a 16-bit or 32-bit interface, the MAC should always align the electrical idle on the parallel interface so that the COM symbol is in the low-order position (*TxDataK[7:0]).*



L0 to L0s

To cause the link to exit the L0s state, the MAC transitions the PHY from the P0s state to the P0 state, waits for the PHY to indicate that it is ready to transmit (by the assertion of *PhyStatus*), and then begins transmitting Fast Training Sequences (FTS). Note, this is an example of L0s to L0 transition when the PHY is running at 2.5GT/s.



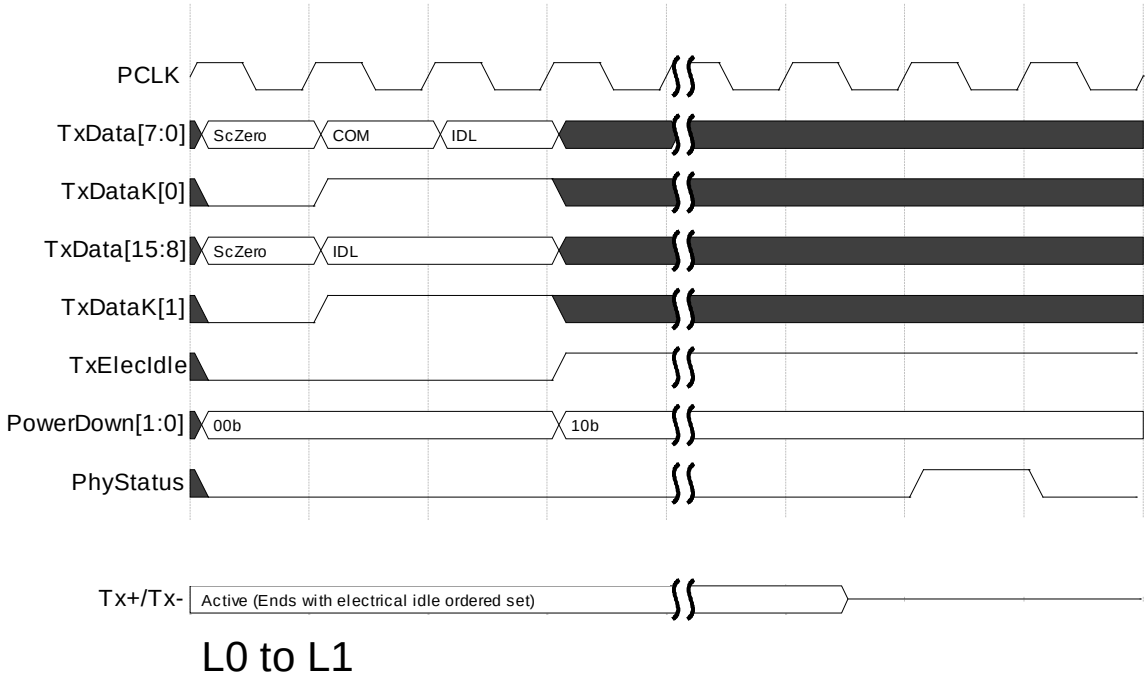
L0s to L0

9.2 Active PM to L1 and back to L0 - – PCI Express Mode

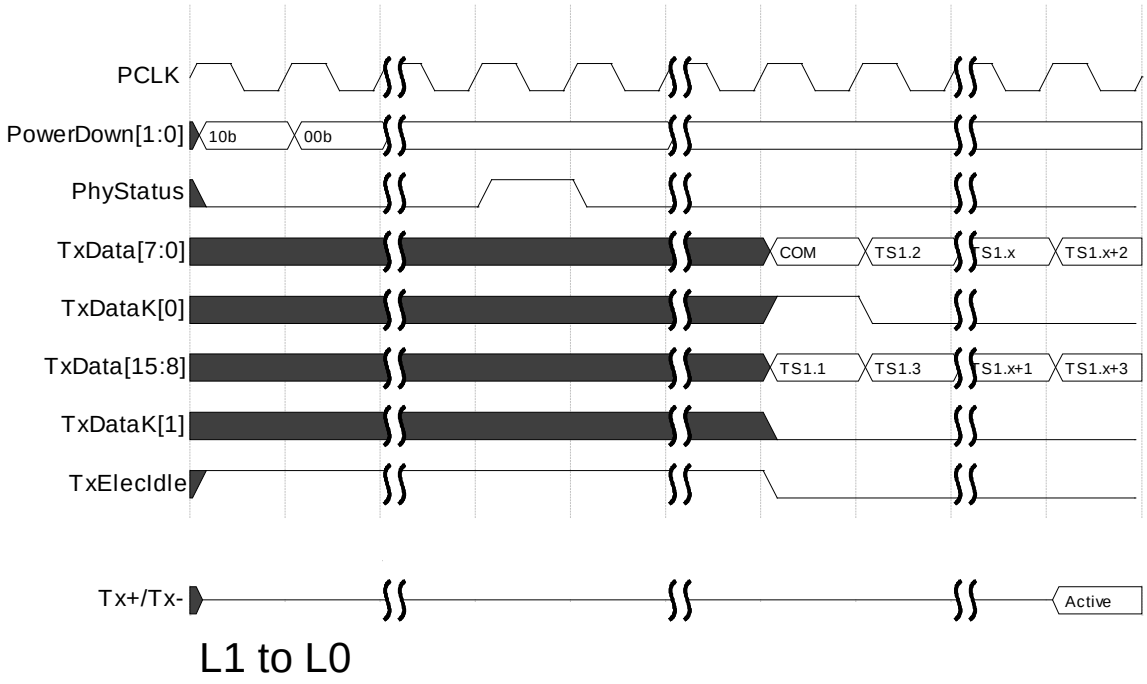
This example shows one way a PIPE PHY can be controlled to perform Active State Power Management on a link for the sequence of the link being in L0 state, transitioning to L1 state, and then transitioning back to L0 state. This example assumes that the PHY is on an endpoint (i.e. it

is facing upstream) and that the endpoint has met all the requirements (as specified in the base spec) for entering L1.

After the MAC has had the PHY send PM_Active_State_Request_L1 messages, and has received the PM_Request_ACK message from the upstream port, it then transmits an electrical idle ordered set, and has the PHY transmitter go idle and enter P1.

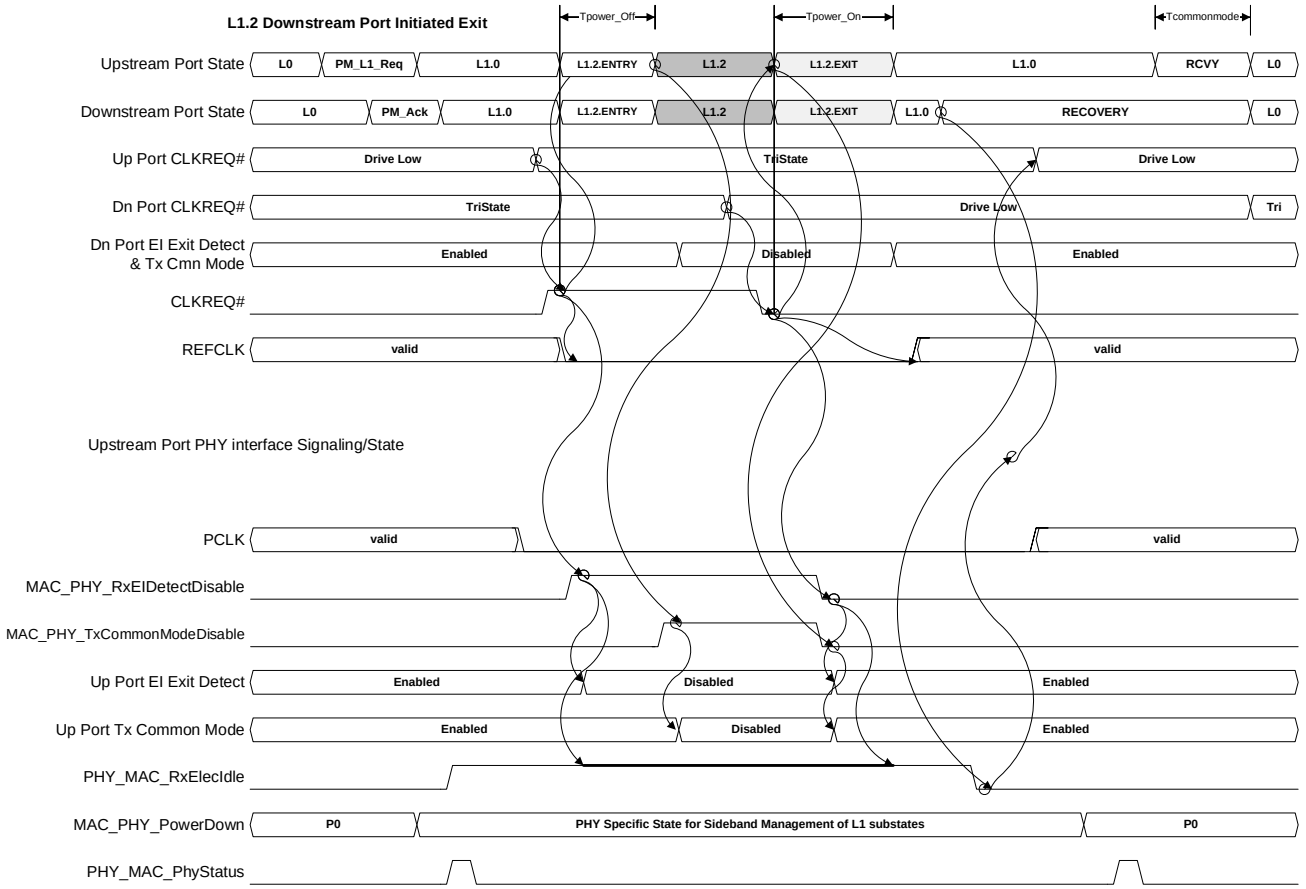


To cause the link to exit the 1 state, the MAC transitions the PHY from the P1 state to the P0 state, waits for the PHY to indicate that it is ready to transmit (by the assertion of *PhyStatus*), and then begins transmitting training sequence ordered sets (TS1s). Note, this is an example when the PHY is running at 2.5GT/s.



9.3 Downstream Initiated L1 Substate Entry Using Sideband Mechanism

Figure 9-1. L1 Substate Management using RxElDetectDisable and TxCommonModeDisable

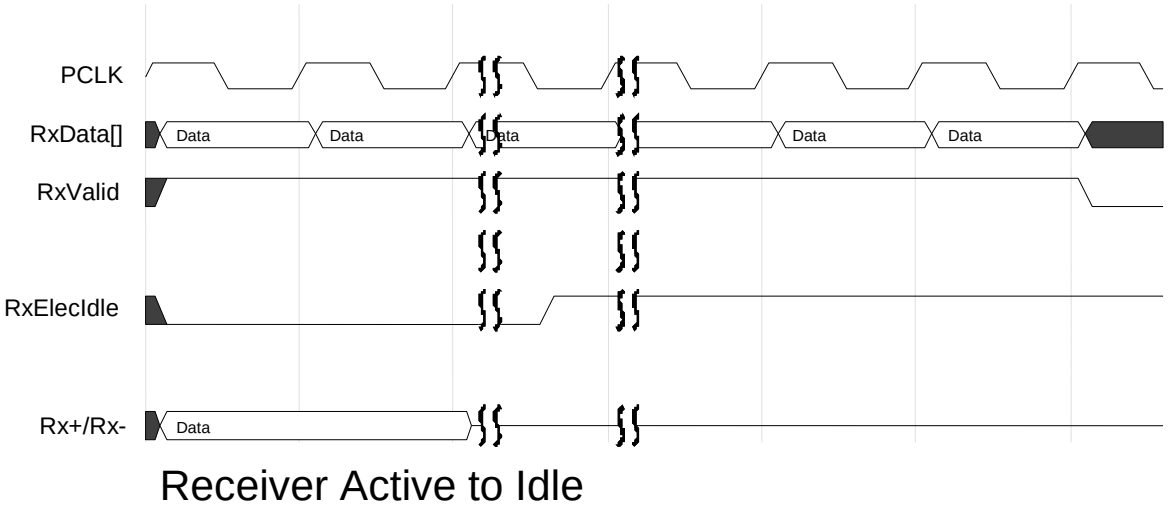


9.4 Receivers and Electrical Idle – PCI Express Mode Example

This section only applies to a PHY operating to 2.5GT/s. Note that when operating at 5.0 GT/s or 8 GT/s signaling rates, *RxElecIdle* may not be reliable. MACs should refer to the PCI Express Revision 3.0 Base Specification or USB 3.0 Specification for methods of detecting entry into the electrical idle condition. Refer to

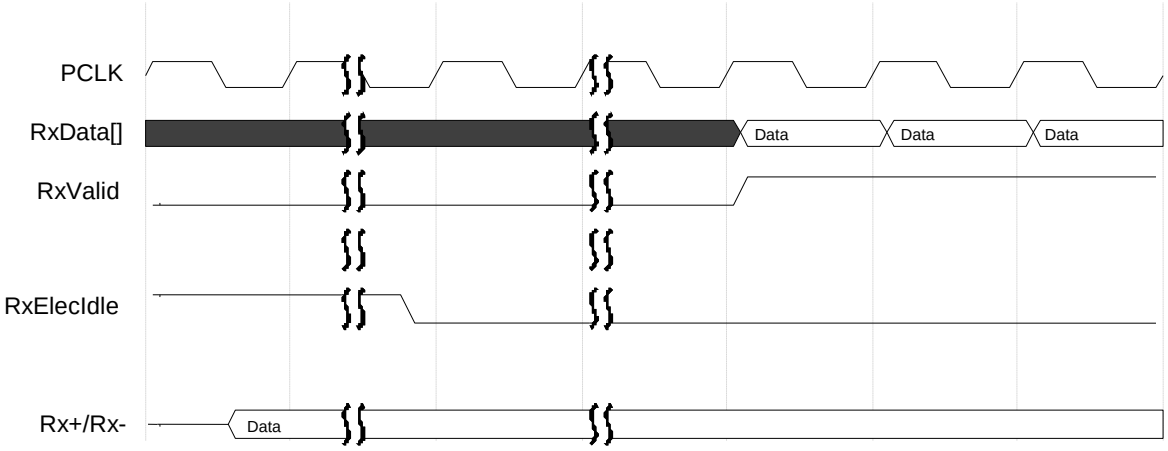
Status Interface for the definition of *RxElecIdle* when operating at 5.0 GT/s. This section shows some examples of how PIPE interface signaling may happen as a receiver transitions from active to electrical idle and back again. In these transitions there may be a significant time difference between when *RxElecIdle* transitions and when *RxValid* transitions.

The first diagram shows how the interface responds when the receive channel has been active and then goes to electrical idle. In this case, the delay between *RxElecIdle* being asserted and *RxValid* being deasserted is directly related to the depth of the implementations elastic buffer and symbol synchronization logic. Note that the transmitter that is going to electrical idle may transmit garbage data and this data will show up on the *RxData[]* lines. The MAC should discard any symbols received after the electrical idle ordered-set until RxValid is deasserted.



Receiver Active to Idle

The second diagram shows how the interface responds when the receive channel has been idle and then begins signaling again. In this case, there can be significant delay between the deassertion of *RxElecIdle* (indicating that there is activity on the *Rx+/Rx-* lines) and *RxValid* being asserted (indicating valid data on the *RxData[]* signals). This delay is composed of the time required for the receiver to retrain as well as elastic buffer depth.



Receiver Idle to Active

9.5 Using CLKREQ# with PIPE – PCI Express Mode

CLKREQ# is used in some implementations by the downstream device to cause the upstream device to stop signaling on REFCLK. When REFCLK is stopped, this will typically cause the CLK input to the PIPE PHY to stop as well. The PCI Express CEM spec allows the downstream device to stop REFCLK when the link is in either L1 or L2 states. For implementations that use CLKREQ# to further manage power consumption, PIPE compliant PHYs can be used as follows:

The general usage model is that to stop REFCLK the MAC puts the PHY into the P2 power state, then deasserts CLKREQ#. To get the REFCLK going again, the MAC asserts CLKREQ#, and then after some PHY and implementation specific time, the PHY is ready to use again.

CLKREQ# in L1

If the MAC is moving the link to the L1 state and intends to deassert CLKREQ# to stop REFCLK, then the MAC follows the proper sequence to get the link to L1, but instead of finishing by transitioning the PHY to P1, the MAC transition the PHY to P2. Then the MAC deasserts CLKREQ#.

When the MAC wants to get the link alive again, it can:

- Assert CLKREQ#
- Wait for REFCLK to be stable (implementation specific)
- Wait for the PHY to be ready (PHY specific)
- Transition the PHY to P0 state and begin training.

CLKREQ# in L2

If the MAC is moving the link to the L1 state and intends to deassert CLKREQ# to stop REFCLK, then the MAC follows the proper sequence to get the link to L2. Then the MAC deasserts CLKREQ#.

When the MAC wants to get the link alive again, it can:

- Assert CLKREQ#
- Wait for REFCLK to be stable (implementation specific)
- Wait for the PHY to be ready (PHY specific)
- Transition the PHY to P0 state and begin training.

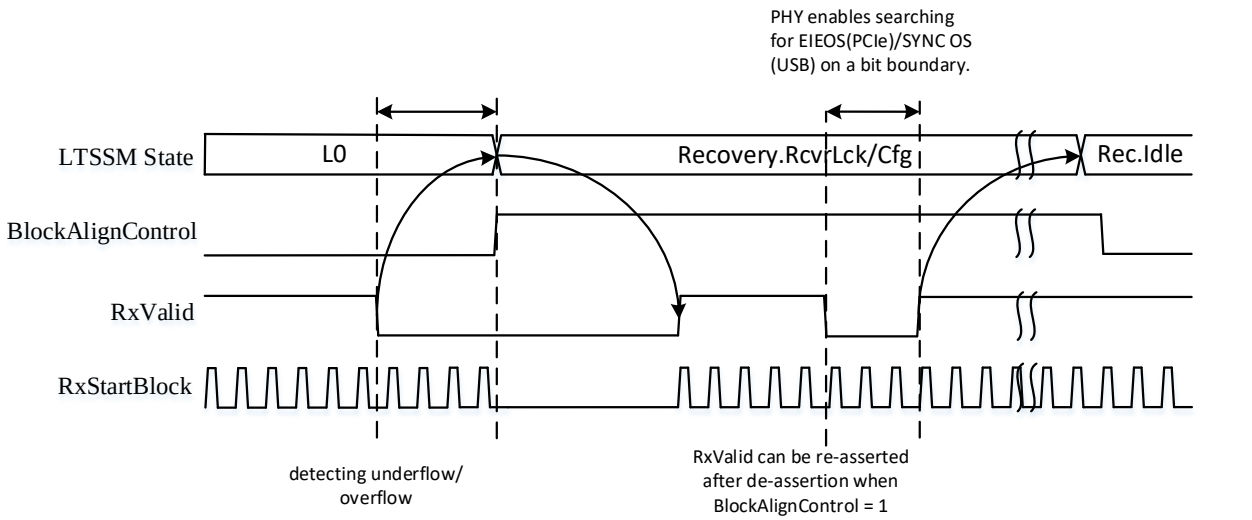
Delayed CLKREQ# in L1

The MAC may want to stop REFCLK after the link has been in L1 and idle for awhile. In this case, the PHY is in the P1 state and the MAC must transition the PHY into the P0 state, and then the P2 state before deasserting CLKREQ#. Getting the link operational again is the same as the preceding cases.

9.6 Block Alignment

Figure 9-2 provides an example of a block alignment sequence using the BlockAlignControl pin. The PHY attempts to do alignment when BlockAlignControl is asserted and the PHY receiver is active.

Figure 9-2. BlockAlignControl Example Timing



Note:
- BlockAlignControl assertion and RxValid deassertion. The PHY will attempt to re-do block alignment.
- The BlockAlignControl assertion does not require the PHY to force the block aligner into the unaligned state.
- There is no requried relationship between the de-assertion of RxValid and RxStartBlock.

9.7 Message Bus: RX Margining Sequence

Figure 9-3 shows an example of an RX margining sequence. The MAC issues a write_uncommitted to address 0x1 followed by a write_committed to address 0x0 to set up the margining parameters and to start margining in the RX Margin Control1 and RX Margin Control0 registers. The PHY issues a write_ack to acknowledge that it has flushed the write buffer. Subsequently, upon processing a change in the 'Start Margin' bit of the RX Margin Control0 register, the PHY issues a write_committed to address 0x0 to assert the 'Margin Status' bit. During the margining process, the PHY periodically issues write_committed transactions to address 0x2 to update the 'Error Count[3:0]' value. The MAC acknowledges receipt of these writes by issuing corresponding write_ack transactions. Finally, the MAC stops the margining process by issuing a write_committed to address 0x0 to deassert the 'Start Margin' bit. The PHY issues a write_ack to acknowledge that it has flushed the write buffer. In response to the 'Start Margin' deassertion, the PHY pushes it's final 'Error Count[3:0]' value to the MAC via a write_uncommitted transaction to the 'RX Margin Status2' register, and then issues a write_committed to assert 'RX Margin Status0.Margin Status'.

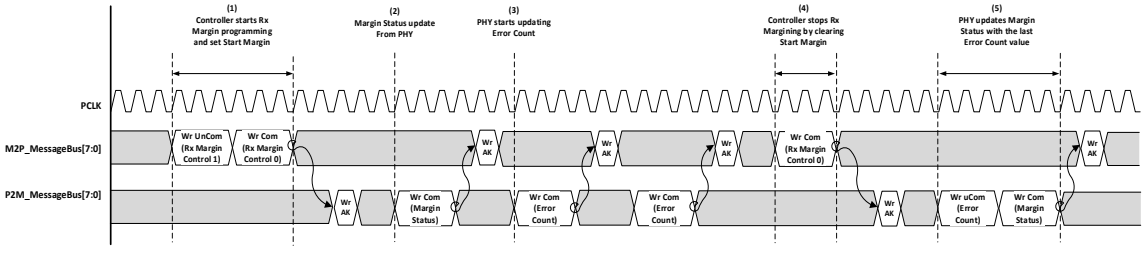


Figure 9-3. Sample RX Margining Sequence

9.8 Message Bus: Updating LocalFS/LocalLF and LocalG4FS/LocalG4LF

Figure 9-4 shows a sequence where LocalFS and LocalLF are updated out of reset and,

subsequently, LocalG4FS and LocalG4LF are updated after a rate change. Note that PhyStatus deasserts only after the write_ack returns for the LocalFS and LocalLF update out of reset. Similarly, the one cycle PhyStatus assertion occurs after the write_ack returns for the LocalG4FS and LocalG4LF update after a rate change. This is one of the rare cases where a dependency between a message bus operation and a dedicated signal exists.

Figure 9-4. LocalFS/LocalLF/LocalG4FS/LocalG4LF Updates Out of Reset and After Rate Change

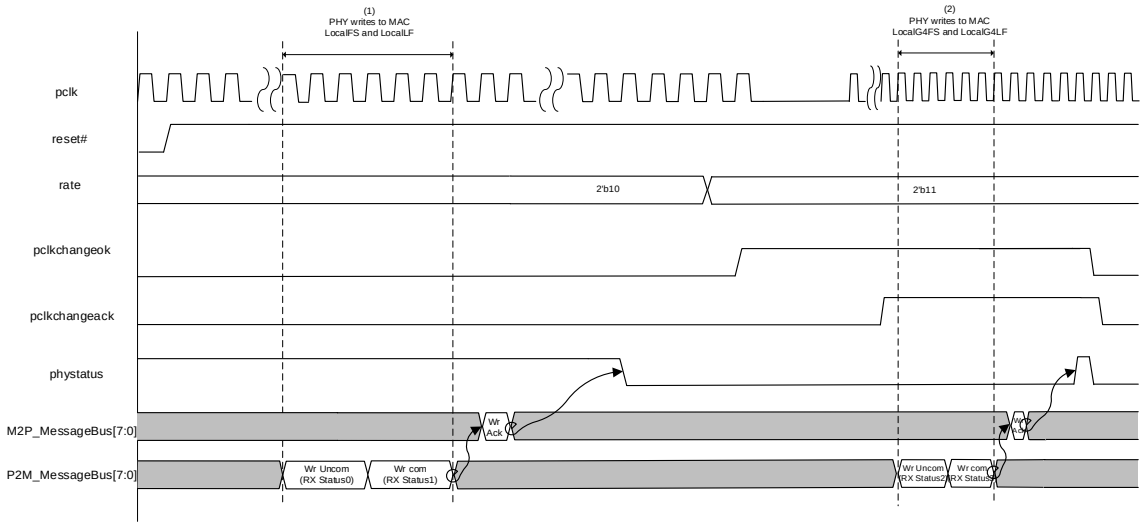
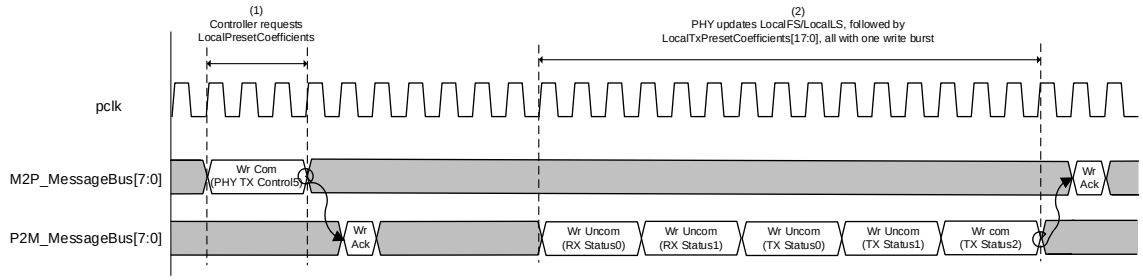


Figure 9-4 shows a sequence where LocalFS and LocalLF are updated in response to a GetLocalPresetCoefficients request where the LocalPresetIndex corresponds to an 8GT/s rate. Note that the LocalFS and LocalLF values must be updated before or at the same cycle as the LocalTxPresetCoefficents are returned.

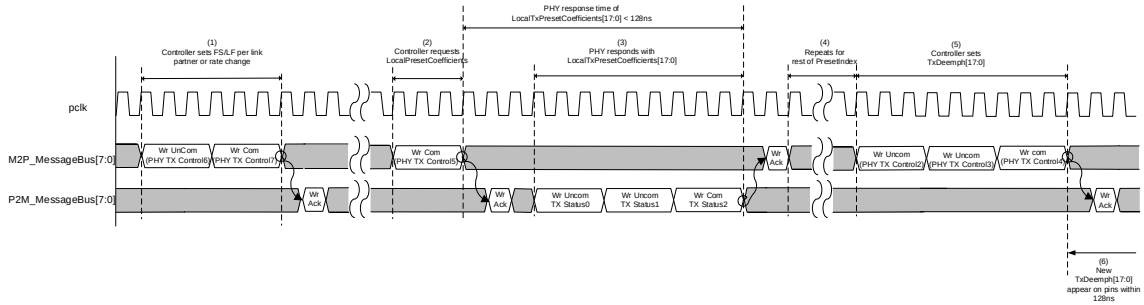
Figure 9-5. LocalFS/LocalLF Update Due to GetLocalPresetCoefficients



9.9 Message Bus: Updating TxDeemph

Figure 9-6 shows a sequence where the MAC makes a GetLocalPresetCoefficients request for one or more values of LocalPresetIndex and the, subsequently, update the TxDeemph value. Note that for every GetLocalPresetCoefficients request, there is a 128ns maximum response time for the PHY to return the LocalTxPresetCoefficients value; this time is shown in the diagram from the end of the second write_committed to the end of the third write_committed. This maximum response time requirement only exists for designs that use just-in-time fetching of GetLocalPresetCoefficients in response to Tx coefficients request from the link partner; designs that fetch ahead of time can circumvent this requirement. Additionally, after the write_committed for TxDeemph, the new TxDeemph value must be reflected on the pins within 128ns.

Figure 9-6. Updating TxDeemph after GetLocalPresetCoefficients Request



9.10 Message Bus: Equalization

Figure 9-7 shows a successful equalization sequence. RxEqInProgress is asserted for the entire duration of equalization. Multiple RxEqEval requests are made during the equalization process corresponding to different coefficient requests to the far end transmitter. When all the RxEqEval requests are complete, RxEqInProcess is deasserted.

Figure 9-7. Successful Equalization

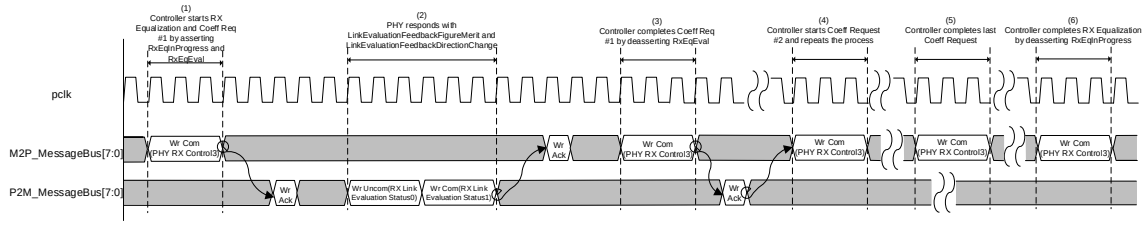


Figure 9-8 shows an equalization sequence where the feedback received indicates an invalid coefficient request for the link partner. Note that the write to assert InvalidRequest must happen before a new request is initiated; the write to deassert InvalidRequest can happen in the same cycle as an RxEqEval request for a new coefficient.

Figure 9-8. Equalization with Invalid Request

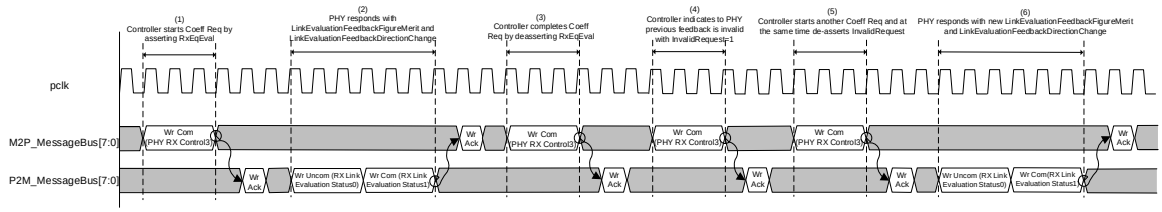


Figure 9-9 shows a sequence where the MAC aborts the RxEqEval request before the link evaluation feedback is returned by the PHY. Figure 9-10 shows a sequence where the MAC aborts the RxEqEval request while the link evaluation feedback is being returned by the PHY, i.e. there is an overlap. In both abort case, the MAC must ignore the feedback value returned by the PHY.

Figure 9-9. Aborted Equalization, Scenario #1

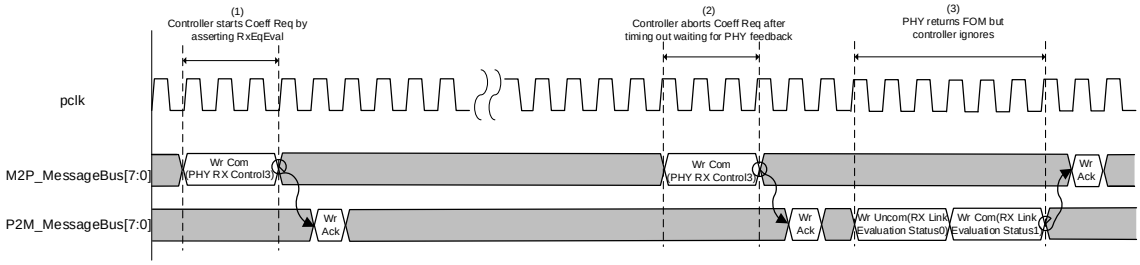
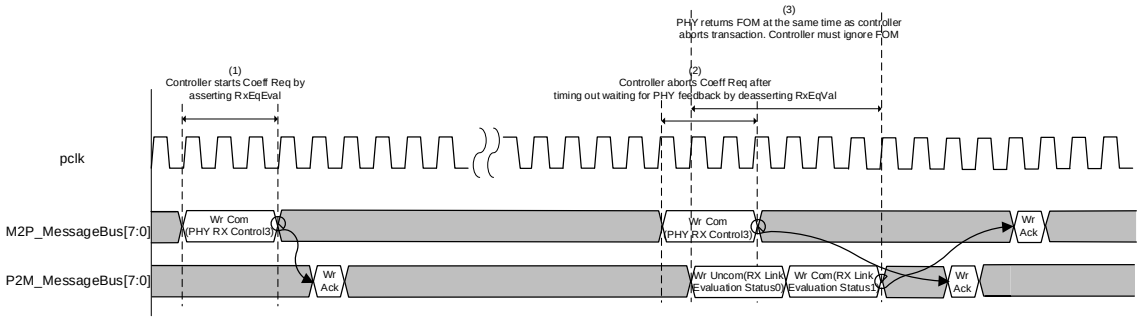


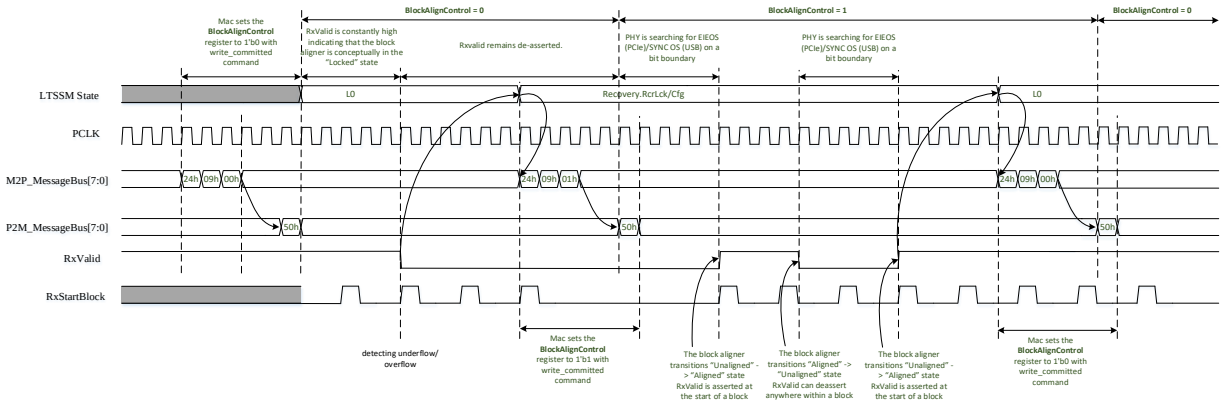
Figure 9-10. Aborted Equalization, Scenario #2



9.11 Message Bus: BlockAlignControl

Figure 9-11 shows a sequence where BlockAlignControl is used to reestablish block alignment after a loss of alignment is detected. This sequence also shows how RxValid transitions during this process.

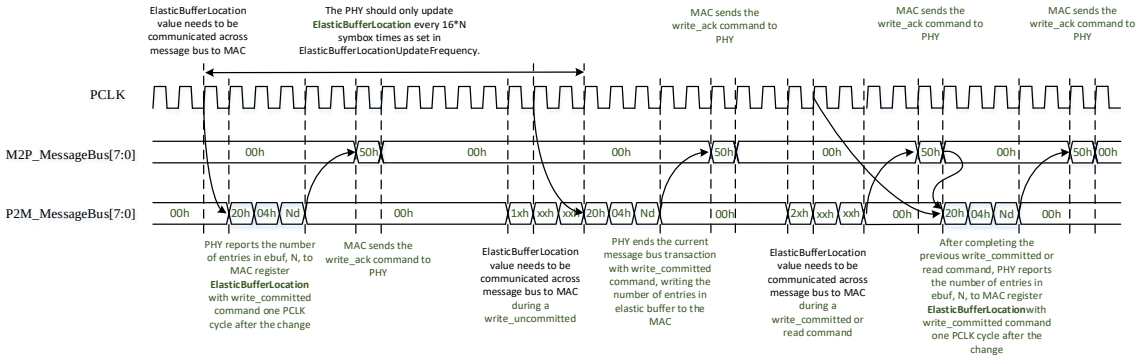
Figure 9-11 Message Bus: BlockAlignControl Example



9.12 Message Bus: ElasticBufferLocation Update

Figure 9-12 shows the update of ElasticBufferLocation across the message bus. The frequency of update across the message bus is controlled by the MAC by setting the value in the ElasticBufferLocationUpdateFrequency register.

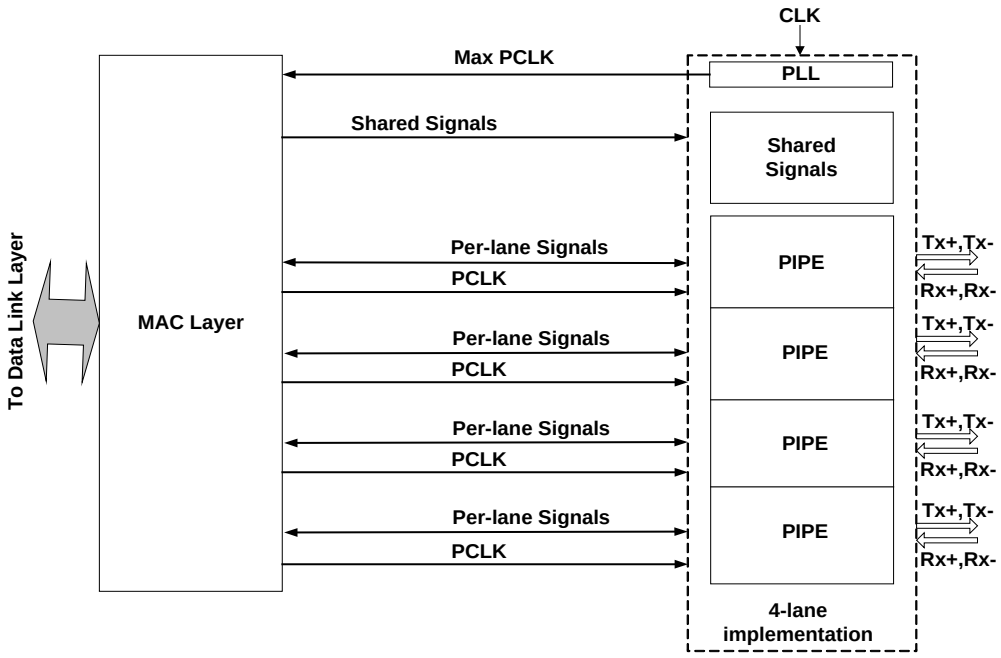
Figure 9-12. Message Bus: Updating ElasticBufferLocation



10 Multi-lane PIPE – PCI Express Mode

This section describes a suggested method for combining multiple PIPEs together to form a multi-lane implementation. It describes which PIPE signals can be shared between each PIPE of a multi-lane implementation, and which signals should be unique for each PIPE. There are two types of PHYs. "Variable" PHYs that are designed to support multiple links of variable maximum widths and "Fixed" PHYs that are designed to support a fixed number of links with fixed maximum widths.

The figure shows an example 4-lane implementation of a multilane PIPE solution with PCLK as a PHY input. The signals that can be shared are shown in the figure as "Shared Signals" while signals that must be replicated for each lane are shown as 'Per-lane signals'.



4-lane PIPE implementation

The MAC layer is responsible for handling lane-to-lane deskew and it may be necessary to use the per-lane signaling of SKP insertion/removal to help perform this function.

	Shared Signals	Per-Lane Signals or Shared Signals	Per-lane Signals
	CLK Max PCLK	EncodeDecodeBypass BlockAlignControl TxSwing TxMargin[2:0] TxDetectRx/Loopback Rate Width[1:0] PCLK Rate[2:0] Reset# TxDataValid PCLK	TxData[], TxDataK[] RxData[], RxDataK[] TxStartBlock TxElecIdle TxCompliance RxPolarity RxValid RxElecIdle RxStatus[2:0] RxDataValid RxStartBlock TxDeemph[17:0] PowerDown[1:0]

			PhyStatus RxPresetHint[2:0] RxEqEval LinkEvaluationFeedbackFigureMerit[7:0] LinkEvaluationFeedbackDirectionChange[7:0] InvalidRequest TxSyncHeader[1:0] RxSyncHeader[1:0] RxStandby RxStandbyStatus FS[5:0] LF[5:0] PHYMode[1:0] SRISEnable Elasticity Buffer Mode TxPattern[1:0] TxOnesZeros RxEqTraining LocalTxPresetCoefficients[17:0] LocalFS[5:0] LocalLF[5:0] LocalPresetIndex[4:0] GetLocalPresetCoefficients LocalTxCoefficientsValid RxEqInProgress RXTermination AlignDetect PowerPreset PclkChangeOk PclkChangeAck ElasticBufferLocation[N:0] AsyncPowerChangeAck M2P_MessageBus[7:0] P2M_MessageBus[7:0]

A MAC must use all “Per-Lane Signals or Shared Signals” that are inputs to the PHY consistently on all lanes in the link. A PHY in “PCLK as PHY Output ” mode must ensure that PCLK and Max PCLK are synchronized across all lanes in the link. A MAC must provide a synchronized PCLK as an input for each lane when controlling a PHY in “PCLK as PHY Input ” mode with no more than 300 ps of skew on PCLK across all lanes.
It is recommended that a MAC be designed to support both PHYs that implement all signals per lane and those that implement the “Per-Lane or Shared Signals” per link. A “Variable” PHY must implement the signals in “Per-Lane Signals or Shared Signals” per lane. A “Fixed” PHY may implement the signals in “Per-Lane Signals or Shared Signals” as either Shared or Per-Lane. A “Fixed” PHY should implement all the signals in “Per-Lane Signals or Shared Signals” consistently as either Shared or Per-Lane.

In cases where a multi-lane has been ‘trained’ to a state where not all lanes are in use (like a x4 implementation operating in x1 mode), a special signaling combination is defined to ‘turn off’ the unused lanes allowing them to conserve as much power as the implementation allows. This special ‘turn off’ signaling is done using the *TxElecIdle* and *TxCompliance* signals. When both are asserted, that PHY can immediately be considered ‘turned off’ and can take whatever power

saving measures are appropriate. The PHY ignores any other signaling from the MAC (with the exception of *Reset#* assertion) while it is 'turned off'. Similarly, the MAC should ignore any signaling from the PHY when the PHY is 'turned off'. There is no 'handshake' back to the MAC to indicate that the PHY has reached a 'turned off' state.

There are two normal cases when a lane can get turned off:

1. During LTSSM Detect state, the MAC discovers that there is no receiver present and will 'turn off' the lane.
2. During LTSSM Configuration state (specifically Configuration.Complete), the MAC will 'turn off' any lanes that didn't become part of the configured link.

As an example, both of these cases could occur when a x4 device is plugged into a x8 slot. The upstream device (the one with the x8 port) will not discover receiver terminations on four of its lanes so it will turn them off. Training will occur on the remaining 4 lanes, and let's suppose that the x8 device cannot operate in x4 mode, so the link configuration process will end up settling on x1 operation for the link. Then both the upstream and downstream devices will 'turn off' all but the one lane configured in the link.

When the MAC wants to get 'turned off' lanes back into an operational state, there are two cases that need to be considered:

1. If the MAC wants to reset the multi-lane PIPE, it asserts *Reset#* and drives other interface signals to their proper states for reset (see section 6.2). Note that this stops signaling 'turned off' to all lanes because *TxCompliance* is deasserted during reset. The multi-lane PHY asserts *PhyStatus* in response to *Reset#* being asserted, and will deassert *PhyStatus* when *PCLK* is stable.
2. When normal operation on the active lanes causes those lanes to transition to the LTSSM Detect state, then the MAC sets the *PowerDown[1:0]* signals to the P1 PHY power state at the same time that it deasserts 'turned off' signaling to the inactive lanes. Then as with normal transitions to the P1 state, the multi-lane PHY will assert *PhyStatus* for one clock when all internal PHYs are in the P1 state and *PCLK* is stable.

11 Appendix

11.1 DisplayPort AUX Signals

Table 11-1. DisplayPort AUX Signals

Name	Direction	Active Level	Description
TxAuxData	Input	N/A	DisplayPort asynchronous transmit data for AUX CH
TxAuxOE	Input	High	DisplayPort asynchronous data output enable for AUX CH. Assertion of this signal must be mutually exclusive with assertion of RxAuxIE.
RxAuxIE	Input	High	DisplayPort asynchronous data input enable for AUX CH. Assertion of this signal must be mutually exclusive with assertion of TxAuxOE.
RxAuxData	Output	N/A	DisplayPort asynchronous data output for AUX CH

DCAux+	Output	Low	Optional: DPRX asynchronous AUX+ pulldown status signal indicates a source is connected (DC voltage)
DCAux-	Output	High	Optional: DPRX asynchronous AUX- pullup status status signal indicates a connected source is powered up (DC voltage)
AuxRxElecIdle	Output	High	Indicates whether differential signaling is detected

